UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA
INGENIERÍA INFORMÁTICA

MOVILIDAD EN IPv6 CON HIP

AUTOR: **Fco. Javier Melero de la Torre**

DIRECTOR: **Carlos Jesús Bernardos Cano**

Dpto. Ingeniería Telemática

Leganés, Mayo 2009

RESUMEN - *ABSTRACT*

En la actualidad, la Internet afronta el reto de responder a una demanda creciente de escalabilidad y prestaciones que va más allá de sus previsiones iniciales. En este contexto, existen varios desarrollos dirigidos a eludir, de una forma u otra, una tradicional restricción de la arquitectura TCP/IP: el acoplamiento entre las conexiones de transporte y la dirección de red. Así, al hablar de movilidad, *multihoming*, virtualización, *clustering*, traducción de direcciones o tránsito de IPv4 a IPv6, encontramos reiteradamente esta limitación que dificulta la persistencia de las conexiones extremo a extremo ante cambios en el nivel de red. HIP (*Host Identity Protocol*) es una propuesta radical en este campo que aboga por resolver el problema introduciendo un nuevo nivel de protocolos en la pila IP, denominado "Nivel de *Host"* (situado entre los de red y transporte), y un nuevo espacio de nombres asociado a él formado por "Identidades de Host". En el presente trabajo, se ha revisado el estado actual de las especificaciones de HIP, se han examinado las primeras implementaciones disponibles y, en el campo de la movilidad IPv6, se han realizado algunas pruebas de funcionamiento, incluyendo una comparativa con MIP (*Mobile IP*).

Nowadays, Internet infrastructure faces the challenge to respond to a growing demand for scalability and functionalities further than its initial forecasts. In this context, there are several developments pointed to elude, one way or another, a traditional architecture restriction in TCP/IP: the coupling of transport connections to network address. Thus, when talking about mobility, multihoming, virtualization, clustering, address translation or transition from IPv4 to IPv6, this constraint repeatedly arise making difficult end to end connections persistency when network level changes take place. HIP (Host Identity Protocol) is a radical approach in this area which claims to solve the problem bringing a new protocol layer into the IP stack, called "Host Layer" (located between those of network and transport), and a new name space linked to it made up of "Host Identities". In this work, ongoing HIP specifications have been reviewed, early implementations have been examined and, in the field of IPv6 mobility, some functional tests have been made, including a comparative with MIP (Mobile IP).

TABLA DE CONTENIDO

INTRODUCCIÓN Y OBJETIVOS DEL PFC

La arquitectura de Internet ha ido evolucionando a lo largo de los años al ritmo de las necesidades de sus usuarios. Ciertos conceptos que no existían originalmente, como la traducción de direcciones, el direccionamiento dinámico o la virtualización, se han introducido para satisfacer una demanda creciente de dimensionamiento y prestaciones. Esto ha supuesto el desarrollo de importantes extensiones al funcionamiento básico del protocolo TCP/IP, aumentando, por tanto, su complejidad y poniendo a prueba sus fundamentos. Veamos algunos ejemplos.

En los últimos años el tamaño de Internet se ha disparado haciéndose necesario el uso generalizado de direccionamiento privado y dispositivos de traducción de direcciones (NAT)[1]. Desde el punto de vista de un servidor, es bastante probable que la dirección y el puerto origen de las peticiones recibidas no sean las mismas que usan sus clientes. Y para un nodo (*host*) que tenga un dispositivo NAT entre él e Internet, le será difícil determinar la dirección y el puerto con que un servidor recibe sus paquetes y tendrá complicaciones para actuar él mismo como servidor.

El direccionamiento dinámico, ideado inicialmente para utilizar de forma más eficiente las escasas direcciones públicas disponibles, ha encontrado otros usos. Los nodos son ahora mucho más móviles de lo que eran hasta hace pocos años y, gracias al direccionamiento dinámico, pueden obtener una dirección topológicamente válida al cambiar de punto de conexión a Internet. Esto permite la denominada portabilidad de un nodo, aunque no una auténtica movilidad; dado que las conexiones TCP incluyen la dirección IP en su información de estado, al cambiar de dirección se perderá cualquier conexión abierta con la dirección antigua.

Pero uno de los mayores cambios habidos en Internet está en sus contenidos, la clase de recursos que los usuarios utilizan y cómo se conectan a ellos. Mientras que en el pasado lo habitual era acceder a un nodo a través de una dirección IP concreta, hoy probablemente estamos más interesados en acceder a un servicio. Deseamos conectarnos con nuestro banco, un periódico o una tienda de comercio electrónico, y poco importa el nodo en que reside dicho servicio.

[1] El acceso comercial a Internet para particulares y pequeñas empresas suele ser de este tipo. El abonado sólo dispone de una IP pública, fija o no, que virtualmente comparten todos los nodos de su red local.

Se ha desarrollado toda una industria sobre el concepto de los llamados proveedores de contenidos y las redes de servicios (*overlay networks*), que hacen un uso extensivo de tecnologías como la virtualización y el *clustering*. En este escenario, la dirección IP de un servidor sólo proporciona una localización efímera para contactar con un determinado servicio.

En resumen, podemos decir que el modelo de direccionamiento de Internet se ha desplazado actualmente hacia un escenario de enlaces dinámicos entre los servicios, los nodos y las direcciones IP. A un nodo se le asigna una dirección IP, según el lugar o el momento, cuando necesita hacer uso de una localización topológica dentro de la red. Un solo nodo puede dar soporte a varios servicios, utilizando una o varias direcciones IP. Y también es cierto al contrario, ya que un determinado servicio puede estar siendo prestado por un indeterminado grupo de servidores.

Algunos trabajo publicados (1) (2) (3) han profundizado en conceptos básicos como son dirección, servicio o nombre, aportando un diagnostico desde el punto de vista de la arquitectura de Internet. Resumidamente, su conclusión es que el conjunto clases de objetos manejados por el modelo de red TCP/IP puede resultar demasiado escaso. O más concretamente, que el espacio de nombres de las direcciones IP se utiliza para demasiadas funciones y sufre una, denominada, sobrecarga semántica. Concretamente, la dirección IP se usa como localizador en la red, como nombre de interfaz, como nombre de nodo y como elemento para la identificación de las conexiones TCP[2].

Para eludir estas dificultades se han planteado numerosas aproximaciones que, como veremos, proponen cambios más o menos drásticos, centrándose algunas de ellas en añadir nuevos identificadores, además de los nombres DNS y las direcciones IP. Podemos pensar que, añadiendo un nuevo espacio de nombres para los extremos lógicos de una conexión, se podrían recuperar las conexiones estables y directas, perdidas como resultado de todos los cambios antes citados en el uso de las direcciones IP. Una de las más recientes y activas propuestas en este campo es el protocolo HIP (*Host Identity Protocol*) (4).

En este trabajo se pretende hacer un estudio sobre el protocolo HIP, el estado de maduración de sus especificaciones e implementaciones, y su uso en el campo de la movilidad en IPv6. El objetivo final que nos proponemos es hacer

[2] Lo mismo sucede con UDP, aunque en este caso el *checksum* de la pseudo-cabecera es opcional, a diferencia de TCP donde es obligatorio. De hecho, es una práctica corriente que los *checksum* UDP estén deshabilitados.

una comprobación operativa del protocolo, es decir, acreditar su capacidad para gestionar de forma eficiente las conexiones de un nodo itinerante. Para ello, utilizando las distintas implementaciones HIP disponibles como soporte de movilidad, se probarán diferentes aplicaciones nativas IPv6 utilizándolas sin modificación alguna. A su vez, se tratará de comparar el funcionamiento de HIP con el de MIP (*Mobile IP*) (5), otra solución de movilidad desarrollada con anterioridad.

La primera parte de esta memoria es una visión general de la problemática a la que el protocolo HIP pretende encontrar soluciones. Esto incluye, la semántica del modelo de red y la introducción del concepto de extremo (*endpoint*) en dicho modelo mediante un nuevo espacio de nombres. A continuación, se hace una clasificación de las principales dificultades que implica la movilidad en las redes IP y las posibles soluciones utilizadas. Por último, se incluye una recopilación de otras aproximaciones propuestas que afrontan estos temas total o parcialmente.

En la segunda parte se hace una descripción de los fundamentos del nuevo espacio de nombres de Identidad de Host (*Host Identity*), y del nuevo nivel de protocolo asociado a él, el Protocolo de Identidad de Host (*Host Identity Protocol,* HIP). Veremos cuáles son las bases de los espacios de nombres actuales y como un nuevo espacio de nombres podría complementarlos. A continuación, se profundiza en sus especificaciones, así como en sus principales extensiones. Por último, se hace una recapitulación de las implicaciones de seguridad y beneficios potenciales.

En la tercera parte se expone el trabajo realizado con las tres implementaciones de HIP que se encuentran ya disponibles. Se comentan los diferentes escenarios de pruebas utilizados para la realización del trabajo y sus ventajas e inconvenientes. Por último, se exponen las pruebas realizadas, incluyendo la operatividad básica del protocolo HIP y su uso con aplicaciones IPv6. La última prueba es una comparación cuantitativa entre HIP y MIP acerca de la eficiencia en la gestión de la movilidad, es decir, el tiempo requerido para restablecer las conexiones tras el movimiento de un nodo.

Por último, en la parte cuarta se resumen las conclusiones de este trabajo y las posibles líneas de trabajo futuras. También, se incluye una breve descripción de las tareas gestionadas en este proyecto y una valoración económica aproximada.

PARTE I: ESTADO DEL ARTE

Ya desde los comienzos de la era Internet, se han ido publicando algunos trabajos (1) (2) que ponían de manifiesto cierta debilidad de su arquitectura, concretamente, que el conjunto clases de objetos manejados por el modelo de red TCP/IP podía resultar demasiado escaso. O dicho de otro modo, que la dirección IP, como único espacio de nombres en Internet, se utiliza para demasiadas cosas y sufre una, denominada, sobrecarga semántica. También se sostiene que los planteamientos iniciales no fueron suficientemente cuidadosos en diferenciar los objetos fundamentales y los nombres de estos, lo que provoca confusión entre las propiedades de esos objetos y las de sus identificadores.

Desde aquellos años, la comunidad de Internet ha hecho frente con éxito a un espectacular crecimiento en tamaño y prestaciones. Pero ese éxito, a su vez, ha planteado nuevos retos y dificultades: la seguridad, la movilidad, el *multihoming*, las redes de servicios, la migración de procesos, etc., y en todos estos casos, las limitaciones semánticas de la arquitectura siguen creando complicaciones.

En la actual arquitectura de Internet existe un reducido número de objetos fundamentales con espacios de nombres asociados; básicamente las direcciones IP y los nombres DNS. Sin embargo, los nombres DNS son algo adyacente a la arquitectura TCP/IP, otro nombre de las direcciones IP más apropiado para las personas y, por lo tanto, redundante. En último término, las direcciones IP son los únicos nombres que, de hecho, están presentes por toda la arquitectura de TCP/IP, y se utilizan para distintas tareas:

- Los *routers* las usan como dato necesario de **enrutamiento** para poder procesar (reenviar) cada paquete recibido.
- Todos los nodos las usan para referirse a las localizaciones topológicas de la red, es decir, los **puntos de conexión**.
- Cada nodo pueda usarlas para nombrar sus propios **interfaces**.
- Los nodos implicados en una comunicación punto a punto las usan como información identificativa de los extremos implicados. Estas son, por tanto, parte de la identificación de las **conexiones TCP** junto al número de puerto, aunque éste sólo sirve como discriminante entre las múltiples conexiones TCP que puede tener un nodo.

Así, surge la impresión de que en una comunicación punto a punto hay implicadas diferentes clases de objetos que deberían poder identificarse de forma individualizada, es decir, tener un nombre propio. Sin embargo, debido probablemente a que fue diseñado para casos de uso más simples de los que hoy en día pretendemos conseguir, el modelo TCP/IP no diferencia estas clases de objetos de forma suficientemente clara, lo que dificulta el modelado de un sistema de comunicación de datos que consiga manejar determinadas situaciones.

La movilidad de un nodo es uno de esos casos que estaba fuera de las expectativas de usuarios y diseñadores hasta hace sólo algunos años. Esta nueva funcionalidad ha ido surgiendo al intentar hacer un uso de la red cada vez más exigente, cuando TCP/IP ya era un estándar muy extendido e Internet una red ampliamente desplegada. Y la movilidad, como veremos, acarrea toda una serie de problemas a distintos niveles que trataremos de examinar a continuación.

UN MODELO DETALLADO DE RED

En una RFC de 1993, J. Saltzer (1) ya proponía un modelo de red flexible basado en objetos y enlaces entre objetos. Dicho modelo incluía las siguientes clases de objetos:

- **Servicios**: las funciones directamente demandadas por los usuarios o clientes.
- **Nodos (*Hosts*)**: las entidades que dan soporte computacional a los servicios, a sus clientes o realizan otras tareas necesarias, como el enrutamiento de los paquetes.
- **Puntos de conexión**: los puertos de la red, es decir, los puntos a los que se conectan los nodos para tener acceso a dicha red.
- **Rutas**: recorridos que discurren entre puntos de conexión atravesando interconexiones y nodos.

Cada una de estas clases de objetos puede tener más de una instancia, lo que implica la necesidad de nombres para identificarlas. Deberá existir, por tanto, un espacio de nombres para cada clase y un enlace entre cada objeto y su nombre.

Además, para el funcionamiento del modelo, deben existir una serie de relaciones entre los objetos de distintas clases. En concreto, para poder enviar un paquete necesitamos: encontrar un nodo que preste el servicio deseado, encontrar un punto de conexión al que esté conectado dicho nodo, y encontrar una ruta que llegue hasta dicho punto de conexión. Lo que significa que deberán existir tres servicios de resolución entre los espacios de nombres definidos para las clases de objetos (de servicio a nodo/s, de nodo a punto/s de conexión y de punto de conexión a ruta/s)

Con este modelo, en palabras de Saltzer, se podría afirmar que:

> *La mayoría de los requerimientos de identificación en una red pueden describirse de forma simple y concisa en términos de enlaces y cambios de enlaces entre las cuatro clases de objetos*

Como ejemplos se proponen los siguientes casos:

- Un servicio dado puede prestarse desde uno o más nodos, y puede necesitar moverse de un nodo a otro sin perder su identidad como servicio.
- Un nodo dado puede estar conectado a uno o más puntos de conexión, y puede necesitar moverse de un punto a otro conservando su identidad como nodo.
- Una pareja dada de puntos de conexión pueden estar conectados por una o más rutas, y puede ser necesario cambiar de una ruta a otra sin afectar la identidad de los puntos de conexión.

Conviene precisar que en los tres casos se incluye la idea de preservar la identidad del objeto. Es decir, que los enlaces entre los objetos y sus nombres deben permanecer estables durante los cambios de enlaces entre objetos. Que un servicio dado, por ejemplo, pueda cambiar de nombre, independientemente de su posible utilidad, probablemente implicaría cambios mayores que el de un simple enlace. Supondría modificaciones en la configuración del software y en su documentación, y debería ser notificado a todos los usuarios. Esta clase de cambios generalmente escaparían del ámbito de este modelo.

Lamentablemente, la arquitectura TCP/IP necesitaría varios cambios para parecerse al modelo de Saltzer, y los cambios son muy difíciles en una red tan enormemente desplegada como Internet. Se hace necesaria, por lo tanto, una solución de compromiso que flexibilice el actual modelo de red y pueda desplegarse de forma más realista.

EL CONCEPTO DE EXTREMO

En otro conocido trabajo de 1999, N. Chiappa (2) argumentaba que es posible mejorar notablemente la flexibilidad de la arquitectura TCP/IP, mediante la introducción de un único objeto con un nuevo espacio de nombres asociado. Este nuevo objeto tendría la misión de asumir algunas de las actuales funciones de la dirección IP, redefiniendo a ésta de forma más restrictiva. Aunque en un principio la elección más obvia para este nuevo objeto podría ser la del nodo, Chiappa lo descarta prefiriendo un concepto más abstracto, el extremo (*endpoint*), basado en la idea de comunicación punto a punto.

Existen ciertas funciones de red, como la seguridad o la fiabilidad, que sólo es posible implementar en un esquema de comunicación punto a punto, es decir, entre el origen primero de los datos y su consumidor final. Dichas funciones no se pueden realizar correcta y completamente sin el conocimiento y la ayuda de las entes situados en los extremos de la comunicación. Los extremos, por tanto, se pueden concebir como las entidades participantes en una comunicación punto a punto, independientemente de los nodos y procesos que puedan estar implicados. En palabras de Chiappa, un extremo es:

> *Cualquier combinación de contexto (state) y/o procesos computacionales que viven y mueren de forma unitaria.*

Esta forma de acción unitaria, o principio de unidad, que caracteriza a los elementos que implementan un extremo es una parte importante de la definición. El principio de unidad (*fate sharing*) establece que el diseño más robusto de un sistema se consigue cuando el estado critico (es decir, la información insustituible sobre el estado de una comunicación punto a punto entre dos entidades) se encuentra junto a las propias entidades punto a punto. O dicho de otro modo, no importaría perder la información de una conexión si la aplicación que usaba dicha conexión también se ha perdido.

De este modo, dejando que la dirección IP siga realizando la función de interfaz, el nuevo concepto de extremo puede asumir los papeles de servicio y nodo del modelo de Saltzer, sin estar limitado por la combinación de hardware y/o software que pudiera dar soporte a dichos nodos y servicios. Así, por ejemplo, un extremo puede incluir varios nodos y responder como un solo servicio. Y un extremo, también, puede constar de un solo nodo con varios interfaces y múltiples servicios.

NECESIDAD DE UN NUEVO NIVEL Y SU ESPACIO DE NOMBRES

En la práctica, incorporar el concepto de extremo supone crear un nuevo espacio de nombres, pero además, un mecanismo de resolución entre los nombres de extremos y las direcciones IP. Y si se decidiera añadir dicho mecanismo, parece preferible hacerlo sólo una vez, en lugar de tener que replicarlo para cada diferente protocolo de transporte. Tampoco parece haber ninguna necesidad que justifique que cada nivel de transporte defina su propio espacio de nombres para extremos. Y además, funcionalmente, un solo extremo debería poder usar varias conexiones de transporte y, por tanto, distintos protocolos de transporte. Así, todo el conjunto podría moverse como un agregado modificando únicamente la relación entre dirección IP y nombre de extremo.

Por todo lo anterior, dentro de la arquitectura TCP/IP, el extremo se desacopla del nivel de red pero no puede ser colocado en el nivel de transporte, haciendo necesario constituir un nuevo nivel, o un subnivel, colocado entre los de red y transporte. La misión de éste sería: mantener los enlaces entre interfaces y extremos, proporcionar mecanismos para la resolución y modificación de esos enlaces, y ocultar tales cambios a los niveles superiores proporcionándoles un identificador invariante. En la Internet actual, los niveles de transporte y red están fuertemente acoplados y ninguno de los dos puede evolucionar de forma separada; así, por ejemplo, las decisiones de diseño sobre IPv6 estuvieron, en su momento, condicionadas por la decisión mantener la compatibilidad con el actual TCP, y no crear el correspondiente “TCPv6”.

Actualmente, hay dos espacios de nombres principales que se usan en Internet: las direcciones IP y los nombres DNS. El DNS proporciona nombres jerárquicamente asignados para nodos y servicios. El correo electrónico, las páginas Web, las direcciones SIP, y muchos otros servicios se sirven de los nombres DNS. Cada jerarquía está delegada por un nivel inmediatamente superior, por lo que no hay anonimato en los nombres DNS.

Por otra parte, las direcciones IP son una mezcla de dos espacios de nombres, los nombres de los interfaces de red de un sistema y los de su localización, entendida ésta como vector indicador de la dirección de enrutamiento, es decir, la información utilizada para enviar los paquetes a su destino. Las direcciones IP, habitualmente, sólo dan nombre a los interfaces de red cuando estos están conectados, y aunque inicialmente tenían un significado de largo plazo, hoy en

día en numerosos casos se usan direcciones efímeras y/o no únicas, es decir, que cada vez que un interfaz se conecta pueden tener asignada una dirección distinta.

Hay, por tanto, tres claras deficiencias en los actuales espacios de nombres. Primera, no se puede gestionar el re-direccionamiento dinámico de forma sencilla. Segunda, no se puede proporcionar anonimato de forma segura y consistente. Y tercera, no existe autenticación ni de los nodos ni de los datagramas.

Un nuevo espacio de nombres para extremos podría utilizarse en operaciones punto a punto independientemente del nivel de red, e incluso a través de distintos niveles de red. Esto permitiría asumir un re-direccionamiento rápido causado por el cambio de ubicación de un servicio, un nodo móvil en tránsito, o una reconfiguración de la red. Si el espacio de nombres estuviera basado en criptografía de clave pública, también podría proporcionar autenticación de servicios. Si este espacio de nombres permite la creación local de identificadores sin exigencia de registro, también podrá proporcionar anonimato.

Algunas características deseables para este nuevo espacio de nombres serían las siguientes:

- El espacio de nombres deberá situarse en el núcleo de IP, entendiendo por éste al componente que se encuentra entre las aplicaciones y la infraestructura de transporte de paquetes.
- El espacio de nombres deberá desacoplar completamente el nivel de red de los niveles superiores. Los nombres deberán reemplazar a las direcciones IP en todos los protocolos de nivel superior al de red, lo que puede requerir cambios en las APIs actuales y, probablemente, otras nuevas.
- Los nombres deberán tener una representación de longitud fija para facilitar su inclusión en las cabeceras de los datagramas y los actuales interfaces de programación.
- El uso del espacio de nombres deberá ser asumible por otros protocolos. Se trata principalmente de un problema de tamaño de paquete, aunque también de capacidad de computación.
- Deberá ser posible la creación local de nombres. Esto podrá proporcionar anonimato al coste de hacer difícil su resolución. Para mejorar ésta, los nombres podrían contener un componente delegado, sin embargo, el papel más probable para una identidad anónima es la de cliente de un servicio,

por lo que su posible resolución resulta poco importante e incluso no deseable.

- Las colisiones de nombres deberán ser evitadas hasta donde sea posible.[3]
- El espacio de nombres debería proporcionar un servicio de autenticación.
- Los nombres deberían ser duraderos, aunque reemplazables en cualquier momento. Los nombres de corto plazo darían como resultado un mantenimiento tedioso de las listas de control de acceso, o requería una infraestructura para el control centralizado de dichas listas.

El espacio de Identidad de Host es un nuevo espacio de nombres orientado hacia estas ideas. El uso de Identidades de Host requiere su propio nivel de protocolo, el Protocolo de Identidad de Host (*Host Identity Protocol*), entre los niveles de red y transporte. Los nombres están basados en criptografía de clave pública para proporcionar un servicio de autenticación. Y con el diseñado adecuado, éste sería capaz de cumplir todos los requisitos anteriormente expuestos.

[3]Se pueden usar las formulas de la paradoja del cumpleaños para estimar la probabilidad de colisión con una población y un tamaño de nombre dados. En general, para un espacio de nombres aleatorios con n bits de longitud, se producirá una colisión después de generar aproximadamente $1.2\sqrt{2^n}$ nombres. Para 64 bits, este número será de unos 5000 millones. Un tamaño de nombre de 64 bits puede ser, por lo tanto, demasiado pequeño para evitar colisiones en una población grande; por ejemplo, en una población de 50 millones existiría una probabilidad de colisión del 1%. Para 100 bits, no se produciría una colisión hasta aproximadamente 2^{50} nombres generados, es decir, habría una probabilidad de colisión del 1% en una población de 250 billones.

EL PROBLEMA DE LA MOVILIDAD IP

Hasta aquí nos hemos centrado en la justificación de un nuevo espacio de nombres para extremos que podría facilitar la movilidad en Internet y que constituye la aportación más original del protocolo HIP. Sin embargo, la movilidad presenta otros aspectos que este nuevo identificador no soluciona por sí mismo y que deben ser examinados. En concreto, dentro de la arquitectura tradicional de Internet la movilidad es causa de cuatro problemas fundamentales a nivel de red:

- **Direccionamiento**: el direccionamiento y enrutamiento está definido de forma jerárquica para mayor escalabilidad. Como resultado, los nodos móviles cuando se conectan a una nueva red se encuentran casi siempre con una dirección topológicamente incorrecta.
- **Gestión de la localización**: si un nodo móvil cambia su dirección IP para solucionar el problema anterior, no podrá ser alcanzado por el resto de la red; a menos que esa nueva información esté disponible para los otros nodos.
- **Gestión de la conexión**: la transición a una nueva dirección puede romper las conexiones activas (flujos) con otros nodos, ya que los protocolos de nivel de transporte utilizan la dirección IP como parte del identificador de conexión. Aún más, los largos periodos de desconexión pueden hacer que las aplicaciones de niveles superiores fallen, incluso si la gestión subyacente del tránsito entre redes funciona correctamente.
- **Seguridad**: los nodos móviles, cada vez que se mueven, deben ser capaces de autenticarse contra aquellos nodos con los que mantiene conexiones abiertas, y mantener o restablecer las asociaciones de seguridad a nivel de red.

El problema del direccionamiento se resuelve habitualmente mediante la asignación dinámica de direcciones, algo factible hace tiempo mediante el protocolo DHCP o la autoconfiguración de IPv6. La localización y la gestión de la sesión, sin embargo, requieren cambios bien en la infraestructura de red, bien en los nodos finales o en ambos al tiempo, mientras que las soluciones a los problemas de seguridad son todavía un tema bastante abierto.

DIRECCIONAMIENTO Y DETECCIÓN DE MOVIMIENTO

Aunque el problema de obtener una dirección IP válida tras un movimiento puede parecer resuelto con la asignación dinámica de direcciones, este aspecto de la movilidad presenta dificultades con la latencia propia de los protocolos disponibles. Estos fueron diseñados específicamente para proporcionar portabilidad de nodos pero no movilidad, la cual exige tiempos cortos de asignación de direcciones. Con el protocolo DHCP, el proceso de configuración IP debe ser iniciado a petición del propio nodo móvil. Y en el caso del protocolo ND (*Network Discovery*) de IPv6, aunque establece el envío periódico de anuncios RA (*Router Advertisement*) por parte del *router* local, la tasa de envío de dichos mensajes es configurable y suele ser demasiado alta. El problema está, por tanto, en la capacidad del nodo móvil de detectar sus propios movimientos lo antes posible para forzar una re-configuración.

Existen trabajos recientes, como el protocolo DNAv6 (6), que tratan de establecer un método normalizado para la detección e identificación rápida del nivel de enlace, ya sea introduciendo cambios en los nodos finales, en la infraestructura de red o en ambos a la vez. En general, para la detección eficiente del punto de conexión a la red se considera necesaria la definición de tres componentes:

- Un método para los nodos que les permita detectar cambios en el nivel de enlace.
- Un método para los nodos que les permita interrogar a los *routers* a fin de identificar el enlace actual (*link indentification*).
- Un método para los *routers* que les permita responder las consultas de los nodos de forma consistente y con un mínimo retardo.

Cualquier procedimiento de cambio de la dirección IP comienza cuando el nodo recibe alguna información, procedente directamente de la red o de otros niveles de protocolos del propio nodo, que le indica que se ha producido un cambio en el nivel de enlace que podría afectar a la validez de la actual configuración del nivel red. Esta clase de información, denominada pista (*hint*), puede consistir en la finalización del procedimiento de acceso del nivel de enlace (por ejemplo, tras un cambio de punto de acceso en una red *wireless*), el vencimiento de temporizadores en el nivel de transporte, o la recepción de un RA inesperado. Dada la variedad de tecnologías de nivel de enlace y posibles casos de uso, este aspecto de la movilidad puede presentar una amplia casuística.

Pero además, una pista pueden implicar la necesidad de un cambio en la configuración IP, o puede que no. El nodo móvil, por lo tanto, debe implementar algún algoritmo de decisión que, en base a su propia información y aquella obtenida interrogando a sus *routers* locales, le permita distinguir entre aquellas situaciones que precisan cambios y las que no. Dicho algoritmo de decisión debe mantener un balance entre rapidez y eficiencia. Debe conseguir que el proceso de cambio de IP se inicie lo antes posible cuando es preciso y, al mismo tiempo, evitar falsos positivos que podrían suponer una importante pérdida de rendimiento (por ejemplo, en un escenario de movimientos frecuentes pero cortos). En cualquier caso, existen siempre dos posibles políticas: el cambio agresivo (*eager switching*), donde prima la rapidez aún a costa de caer en más falsos positivos, y el cambio perezoso (*lazy switching*), que intenta limitar el número de cambios a los estrictamente necesarios empleando más tiempo en las comprobaciones.

GESTIÓN DE LA LOCALIZACIÓN

Para la localización, hay varias soluciones posibles que se centran en cómo informar al nodo corresponsal sobre los movimientos del nodo móvil. Existen tres opciones:

- **Modo directo**: notificando directamente al nodo corresponsal la nueva localización del nodo móvil.
- **Modo indirecto**: depositando la información sobre localización en la infraestructura de red, de modo que pueda consultarse según se necesite.
- **Modo transparente**: utilizando infraestructura de red de tipo proxy (como el home agent de Mobile IP) que reenvíe el tráfico del nodo móvil a su última localización conocida.

El método transparente tiene una ventaja importante sobre los demás, y es que puede mantener al nodo corresponsal ajeno al hecho de que el otro extremo de la comunicación se está moviendo. Es decir, el único extremo de la conexión que necesita modificaciones en su pila TCP/IP es el nodo móvil, pudiendo comunicarse con cualquier otro nodo. Sin embargo, este enfoque tiene una desventaja que en la práctica se ha revelado como importante. Suele darse el caso de que un nodo móvil desee conectarse con otro nodo muy próximo a él, mientras que su proxy se encuentra a una gran distancia. Esto acarrea un pobre rendimiento difícil de justificar de cara al usuario.

Por otro lado, el modo directo, tiene la ventaja de ser más sencillo y eficiente, pero no es eficaz en todos los casos y es necesario el apoyo de otro método en determinadas ocasiones. Por ejemplo, en el caso de una comunicación entre dos nodos móviles podría ocurrir que ambos nodos se movieran al mismo tiempo, haciendo imposible la actualización de su localización de modo directo. En este caso hace falta un método de respaldo, indirecto o transparente, que permita a ambos nodos volver a encontrarse. Además, el método directo no sirve para iniciar una conexión con un nodo móvil cuando este se encuentra en itinerancia, ya que, el corresponsal hasta ese momento no ha recibido actualizaciones sobre su localización. Parece inevitable, por tanto, utilizar inicialmente el método indirecto o el transparente, aunque a continuación se pase al modo directo.

GESTIÓN DE LA CONEXIÓN

Un nodo móvil se caracteriza por que puede mantener abiertas sus conexiones TCP mientras se mueve por la red. Esto no resulta fácil con la actual arquitectura TCP/IP, ya que las conexiones se identifican mediante la cuádrupla [IP origen, Puerto origen, IP destino, Puerto destino]. Para enviar y recibir paquetes desde un punto distinto de la red será necesaria una nueva dirección IP, pero si cambia la dirección de cualquiera de los extremos los paquetes no serán reconocidos como pertenecientes a la misma conexión. Para sortear esta dificultad se han propuesto diferentes técnicas.

Una de ellas se basa en el uso de una dirección IP invariante para establecer las conexiones TCP. Cuando los paquetes de tales conexiones no pueden ser encaminados directamente se utiliza el encapsulado o, en el caso de IPv6, ciertas extensiones de cabecera (*routing header, destination option headers*) para cursarlos. Así, el nodo móvil debe utilizar dos direcciones IP, una circunstancial que identifica el interfaz y otra persistente que identifica al nodo, es decir, conceptualmente se usan clases de objetos distintas con nombres distintos, aunque pertenecientes a un mismo espacio de nombres. El hecho de que el identificador del nodo siga siendo una dirección IP pública, aunque con otra semántica, hace que cambios topológicos en el nivel de red, como un cambio de proveedor de servicio o una re-numeración, obliguen al nodo móvil a mudar su identidad.

Hay otra forma de solucionar esta dificultad en particular, si no nos atenemos únicamente al nivel de red. Se puede cambiar la definición del protocolo TCP para permitir que una conexión dada pueda cambiar los valores de su cuádrupla

identificativa, permitiendo así la movilidad. Esto implica cierta complejidad y nos deja, además, sin un nombre invariante para la conexión. Pero su principal desventaja está en que es necesario idear un mecanismo similar para cada protocolo de nivel de transporte, siendo preferible un único mecanismo genérico.

Pero existen aún más dificultades en la gestión de la sesión si la movilidad se entiende en un sentido más abstracto, es decir, considerando el concepto de extremo (*endpoint*) propuesto por Chiappa. Así, nos encontramos con la posibilidad de procesos móviles, nodos *multihomed*, redes de servicios, o automatismos de re-numeración de redes que pueden causar problemas más sutiles.

Por ejemplo, consideremos un proceso móvil que tiene una conexión TCP abierta en el puerto X. Este se encuentra en un nodo con dirección A y desea moverse a otro con dirección B. Aún en el caso de conseguir preservar la conexión abierta mediante algún mecanismo de movilidad de nodo, hay problemas que no están resueltos. Primero, si hay más de una conexión entre el nodo A y un tercero N, éste último podría encontrarse con que, aparentemente, el nodo A se ha movido y al mismo tiempo permanece quieto. Y segundo, si hay un proceso en el nodo B que ya está usando el puerto X, se producirá una colisión en dicho puerto que precisará un procedimiento de arbitraje.

Así, la falta de diferenciación entre clases de objetos conceptualmente diferentes, el interfaz y el nodo (o el extremo, en último término) junto con sus espacios de nombres, provoca dificultades que no se pueden resolver de forma sencilla con la actual arquitectura TCP/IP

SEGURIDAD

La ausencia de un espacio de nombres que identifique de forma única a un nodo ha supuesto problemas de diseño en los protocolos ESP[4] y AH[5] de IPsec (7). Estos protocolos deberían ligar sus asociaciones de seguridad a un nombre de nodo distinto de la dirección IP o el nombre DNS ya que ambos pueden variar (por ejemplo, cuando el nodo cambia de localización) mientras que las asociaciones de seguridad podrían permanecer activas. Otra ventaja de un espacio de nombres independiente de la topología de la red es que ESP y AH

[4] *Encapsulating Security Payload.*
[5] *Authentication Header.*

podrían usar dichos nombres para las asociaciones de seguridad que atraviesan dispositivos NAT. En ausencia de un nombre persistente en la arquitectura de Internet, para caracterizar las asociaciones de seguridad actualmente se utiliza la dirección IP; lo que supone un defecto en la arquitectura y no una funcionalidad.

En la actualidad hay varios sistemas que pueden utilizarse para proporcionar seguridad a las comunicaciones IP, como son: SSH, IPsec, SSL, TLS o PGP. Todos ellos, para conseguir una mayor protección, además de cifrar los datos identifican los extremos con claves públicas verificables. Parece razonable pensar que, si se introdujera un nuevo espacio de nombres, estos deberían incluir una funcionalidad similar.

Por otro lado, la movilidad y el *multihoming* están inexorablemente entrelazados con los temas de seguridad. Una solución inadecuada será muy propensa a sufrir diversos ataques sobre los mensajes de actualización de localización usados para notificar los cambios de dirección. Los ataques de denegación de servicio (DoS – *Denial of Service*) y secuestro de dirección IP son problemas obvios, ya que la propiedad de una dirección IP resulta difícil de probar.

TRABAJOS RELACIONADOS

Existe un gran número de trabajos que inciden en los principales aspectos tratados por HIP, como son: la movilidad, el *multihoming*, la identidad, la seguridad y la interconexión entre diferentes espacios de direccionamiento. A continuación, se citan brevemente algunos de ellos

MIP

Mobile IP (MIP) (8) (5), es una solución de nivel red para la movilidad desarrollada durante los últimos años por el IETF. Necesita de una infraestructura adicional, conocida como ***Home Agent*** (HA), diseñada para retransmitir los paquetes entre el nodo móvil y su red origen (*home network*). El nodo móvil tiene asignada una dirección IP pública en la red origen (*home address,* HoA) que utilizará como identificador invariante para establecer todas las conexiones de transporte y comunicarse con otros nodos. Cuando el nodo móvil cambia su punto de conexión a la red actualiza a su agente enviándole la nueva dirección disponible (*care-of address*, CoA), de modo que éste puede interceptar los datagramas enviados al nodo móvil y retransmitírselos. Se logra, así, un punto estático a través del cual intercambiar paquetes con el nodo móvil sin tener que añadir más sobrecarga de enrutamiento global. Además, Mobile IP puede generalizarse para incluir la movilidad de redes.

El modo de localización transparente utilizado permite al nodo móvil mantener su conectividad con los nodos que no lo implementan, pero obliga a los paquetes a realizar un recorrido triangular que puede acarrear ineficiencias; aunque el nodo móvil y su corresponsal se encuentren próximos, el HA implicado puede encontrarse muy alejado. Para solucionar este problema y mejorar la latencia, la fiabilidad y reducir la posible congestión del HA*,* se ha incluido una extensión significativa al planteamiento básico conocida como **optimización de ruta** (*route optimization,* RO). Esta técnica permite que, una vez establecida la comunicación entre el nodo móvil y su corresponsal, estos puedan cortocircuitar la conexión dejando fuera al HA. El nodo corresponsal deberá ser capaz de gestionar las conexiones con el nodo móvil a través de un túnel contra su CoA, es decir, deberá implementar algunas funciones de MIP, perdiéndose así la mejor ventaja del modo transparente.

La forma de funcionar de Mobile IP consiste, por tanto, en usar un túnel dentro de la topología de Internet para simular la presencia de una dirección IP en un

lugar, cuando realmente está en otro pero, en cualquier caso, siempre debe haber una ruta disponible hacia la HoA del nodo móvil. Aunque en IPv6 esto no debería ser un problema, en IPv4 no siempre es posible por sus limitaciones de direccionamiento.

Aún más importante es el hecho de que el nodo móvil, aunque permanezca conectado a la red, no puede superar la pérdida de su HA. Dado que su identidad recae sobre la HoA, si el HA se desconecta ya no recibirá ninguna conexión entrante y las conexiones establecidas no se podrán restablecer tras un nuevo movimiento.

SCTP

Muchos de los problemas planteados hasta ahora tienen que ver con el acoplamiento de los niveles superiores con el nivel de red, como puede ser el uso de la dirección IP en la pseudo-cabecera TCP. El protocolo SCTP[6] (9), una alternativa a TCP, usa la dirección IP de una forma más dinámica como identificador de los extremos de una conexión. En el protocolo TCP éstos se nombran por su dirección IP, habiendo exactamente dos direcciones, una por cada extremo. SCTP permite la existencia de múltiples direcciones por cada extremo, de forma que facilite la redundancia en aplicaciones que exigen alta disponibilidad. Sin embargo, también es posible modificar las conexiones mientras un nodo se mueve de un lugar a otro, o mientras sus direcciones cambian debido a una re-numeración (o cualquiera que sea el motivo). El IETF ha trabajado para incluir una nueva capacidad en el protocolo SCTP, que le permite a un extremo modificar el conjunto de direcciones utilizadas para una conexión.

Esta idea tiene tres limitaciones. Primera, sólo beneficia a los nodos que implementan SCTP, y por tanto precisa ser desplegada. Segunda, los extremos carecen de un identificador estable, lo que trae ciertas dificultades de seguridad. Estas pueden mitigarse con el uso de mecanismos adicionales, como por ejemplo claves ad hoc, que veremos a continuación. Y tercera, dado que SCTP no cuenta con un ningún tipo de mecanismo donde intercambiar información de localización, este protocolo no puede manejar determinados casos particulares, como cuando se intenta abrir una nueva conexión con un nodo en itinerancia o cuando ambos extremos se mueven a la vez.

[6] *Stream Control Transmission Protocol.*

MOBIKE

El protocolo IKEv2[7] (10) se utiliza para proporcionar autenticación mutua entre los extremos de una conexión, así como para el establecimiento y gestión de las Asociaciones de Seguridad (SA) de IPsec. En el protocolo IKEv2 básico, las SA se crean implícitamente entre las direcciones IP de cada nodo en el momento de establecerse. Tales direcciones se usan a continuación como direcciones externas del túnel, y con las actuales especificaciones no es posible cambiarlas. Aunque es posible afrontar el problema de la movilidad creando una nueva SA con cada cambio de dirección del nodo móvil, esto resulta problemático en determinados casos. Establecer una nueva SA puede requerir la intervención del usuario para la autenticación, y tener un coste excesivo en tiempo de cálculo.

Mobile IKEv2 (MOBIKE) (11) es una extensión del protocolo IKEv2 para la movilidad y el *multihoming*, que permite el cambio de la dirección IP asociada con las SA de IPsec en modo túnel. Así, un cliente VPN móvil puede usar MOBIKE para mantener activa la conexión con su pasarela aunque cambie su dirección IP. De modo similar, un nodo *multihomed* puede usar MOBIKE para conmutar su tráfico hacia un interfaz diferente si el actual deja de funcionar.

Aunque MOBIKE puede soportar la movilidad en ambos extremos de la conexión, no incorpora ningún mecanismo de localización para el caso del movimiento simultáneo de ambos nodos, o la busqueda de un nodo en itinerancia. También, MOBIKE soporta que ambos nodos sean *multihomed* aunque sólo pueden usarse un par de direcciones por SA. El balanceo de carga y el *clustering* quedan, por tanto, fuera de sus posibilidades. Futuras extensiones podrían incluir nuevos casos de uso, como la movilidad en modo transporte, o como mecanismo de seguridad para el protocolo SCTP.

RSIP Y MIDCOM

Hay dos enfoques relacionados que buscan como aunar espacios de nombres que entran en conflicto. Uno de ellos, la IP de Ámbito Especifico (RSIP[8]) (12), permite la asignación temporal de direcciones de cierto espacio a un nodo que pertenece a otro espacio distinto, de manera parecida a como funciona el préstamo de direcciones en el protocolo DHCP. En esta

[7] *Internet Key Exchange version 2.*
[8] *Realm Specific IP.*

arquitectura, una pasarela RSIP reemplaza a la pasarela NAT y los nodos deberán implementar también RSIP para poder hacer uso de ella tunelizando el tráfico.

El beneficio de RSIP consiste en que el extremo que recibe el préstamo sabe qué dirección se le asigna, de modo que puede, llegado el caso, propagar esa información a otros nodos. Esto elimina la necesidad de pasarelas de nivel de aplicación (ALG[9]), como le ocurre al actual NAT con, por ejemplo, el protocolo FTP.

Como los nodos RSIP toman prestadas las direcciones públicas deberán devolverlas tan pronto como les sea posible, o la pasarela tendrá que denegar el servicio a otros nodos. Para hacer un mejor uso de tan escaso recurso, una implementación RSIP necesitará procesar el tráfico no sólo en función de la IP de destino, sino también en función de información del nivel de aplicación. Por ejemplo, un nodo interno que sólo desea navegar por la web probablemente no necesitará una IP externa. Además, la pasarela puede prestar la misma dirección IP a varios nodos al mismo tiempo con la condición de limitar su uso a puertos concretos. Por todo esto, el problema con RSIP está probablemente en la complejidad de las decisiones de enrutamiento que el nodo y la pasarela deben tomar y que requieren, al menos, cierta configuración.

Por otro lado, MIDCOM[10] (13) es una arquitectura para la comunicación a través de un dispositivo intermedio, como un NAT o un firewall. Aunque se trata de una aproximación parecida a RSIP, en este caso no se tuneliza el trafico, sino que se añade un agente entre el nodo y el dispositivo intermedio, de modo que el primero entienda, y en su caso controle, la transformación que realiza el segundo. Así, cuando un NAT o una web-cache traducen direcciones de un espacio a otro, MIDCOM permite al nodo final conocer dicha traducción. MIDCOM está pensado para aplicaciones concretas y particularmente complejas, de modo que libera a los dispositivos intermedios de implementar pasarelas ALG y mantenerlas actualizadas.

Sin embargo, ninguno de estos dos enfoques resuelve el problema de cómo descubrir los dispositivos intermedios, y además tienen problemas en el caso de que haya varios en el camino. Tampoco proporciona una forma de identificación univoca y duradera para los nodos situados detrás de un NAT. En definitiva, ambos planteamientos son una evolución de los dispositivos

[9] *Aplication Layer Gateways.*
[10] *Midllebox Comunication.*

intermedios que mejora su funcionamiento respecto al nivel de aplicación, pero no parece una solución a los problemas planteados a largo plazo.

IPNL Y TRIAD

Se han hecho varias propuestas de investigación sugiriendo el uso de los Nombres de Dominio Completos (FQDN) como espacio de nombres para la movilidad y/o enrutamiento a través de varios ámbitos de enrutamiento (*routing realms*). Nuevas arquitecturas basadas en NAT, como TRIAD[11] (14), o IPNL[12] (15), soportan enrutamiento por nombres, y recuerdan a HIP porque introducen un nivel de protocolo adicional entre los de red y transporte.

Aparte de las implicaciones arquitecturales que suponen el uso de nombres de dominio en lugar de identificadores de nodo criptográficos, existe la posibilidad de combinar HIP con IPNL para proporcionar mayor seguridad a este enfoque.

PBK

Las PBK[13] o claves ad hoc (16), son identificadores temporales utilizados para validar un extremo determinado de una comunicación. En lugar de intentar construir una infraestructura para validar la identidad de los extremos, el único propósito de las PBK es el de conseguir que dos nodos que establecen una comunicación puedan continuarla a lo largo del tiempo, con la certeza de que a su finalización los extremos de dicha comunicación seguirán siendo los mismos que eran al comienzo. Así, incluso si algún extremo cambiase de dirección IP, éste podría volver a validarse frente al otro extremo.

Las PBK no hacen afirmación alguna sobre quienes son realmente los participantes de una comunicación. No hacen uso de ninguna clase de Infraestructura de Clave Pública (PKI[14]), sino que son intrínsecamente efímeras y su tiempo de vida es el mismo que el de la comunicación para la cual fueron creadas. Las PBK tienen la forma de parejas de claves pública/privada generadas bajo demanda. Cuando un nodo quiere validarse frente a otro, éste firma criptográficamente un bloque de datos con su clave privada y dicha firma

[11] *Translating Relaying Internet Architecture Integrating Active Directories.*
[12] *IP Next Layer.*
[13] *Purpose Built Keys.*
[14] Public Key Infrastructure.

se verificará por el otro extremo mediante la clave pública asociada. La clave pública, por tanto, se convierte en la propia identidad del nodo.

Las PBK pueden utilizarse en diferentes niveles de la pila de protocolos; por ejemplo, pueden intercambiarse mediante una extensión de la cabecera IPv6 o ser usadas por un determinado protocolo de nivel de aplicación.

PARTE II: HOST IDENTITY PROTOCOL (HIP)

Internet tiene dos importantes espacios de nombres de ámbito global: el espacio de direcciones IP, y el conjunto de los nombres DNS. Estos dos espacios de nombres tienen una serie de características y crean una serie de abstracciones que han permitido a Internet ser lo que hoy en día es. Pero también tienen debilidades. Básicamente, que al ser todo lo que tenemos, intentamos hacer con ellos demasiadas cosas a la vez. Así, la sobrecarga semántica y las funcionalidades añadidas han convertido a estos espacios de nombres en algo complicado de manejar en determinadas circunstancias.

En este apartado comenzaremos examinando el **Espacio de Nombres de Identidad de Host (*Host Identity Namespace*)** que viene a llenar un hueco importante entre los espacios de nombres IP y DNS. Fue presentado por primera vez al IETF por Robert Moskowitz en el año 2001 y actualmente ha alcanzado el status de RFC (4). Es un espacio formado por **Identificadores de Host (*Host Identifier*, HI)** de naturaleza criptográfica que representan a las respectivas **Identidades de Host**. Es decir, la Identidad de Host se refiere a un ente abstracto singularmente identificado, mientras que el HI es un determinado patrón de bits utilizado en el proceso de identificación. Cada nodo tendrá al menos una Identidad de Host, aunque habitualmente tendrá más de una, y cada Identidad de Host identificará de forma única a un solo nodo, o sea, dos nodos nunca tendrán la misma Identidad de Host. Así mismo, una Identidad de Host puede publicarse (por ejemplo, en el DNS) o permanecer anónima, y los sistemas cliente pueden disponer de identidades de ambas clases.

Aunque los Identificadores de Host podrían usarse con otros sistemas de autenticación como IKE, la arquitectura propuesta introduce un nuevo protocolo, llamado **Protocolo de Identidad de Host** (***Host Identity Protocol,*** **HIP**) (17), y un nuevo intercambio criptográfico conocido como **Intercambio Básico** (***Basic Exchange,*** **BEX**). El protocolo HIP proporciona un mecanismo para establecer una relación de confianza entre sistemas, mejora la movilidad, el *multihoming*, la reorganización dinámica de IPs, ayuda a la traducción o transición entre protocolos de red, y es resistente a determinados tipos de ataques de negación de servicio (*Denial of Service,* DoS).

Con el uso de HIP, la carga real del tráfico entre dos nodos HIP estará preferiblemente, aunque no de forma necesaria, protegida mediante IPsec. Las HIs se utilizarán para crear las necesarias **Asociaciones de Seguridad IPsec**

(***Security Asociation,* SA**) y para autenticar a los nodos. Cuando se utiliza IPsec, el tráfico real de paquetes IP no se diferencia en absoluto del habitual tráfico IP protegido por IPsec.

EL ESPACIO DE NOMBRES HIP

En el Espacio de Nombres de Identidad de Host, cada elemento HI representa un nombre de ámbito global y estadísticamente único para nombrar cualquier sistema con una pila de protocolos IP (IP *Stack*). Una identidad está asociada habitualmente, aunque no limitada, a una pila IP mientras que un sistema puede tener múltiples identidades, unas conocidas (públicas o publicadas), y otras anónimas. Un sistema puede afirmar su propia identidad, o puede usar a un tercero como autoridad de certificación, de forma similar a como lo hace el DNS seguro (DNSSEC), PGP, o X.509 para refrendar la identidad.

La estructura que define la Identidad de Host es aquella compuesta por una pareja de claves pública y privada. El nombre que representa una Identidad de Host en el espacio de nombres de Identidades de Host, es decir, el Identificador de Host (HI), es la propia clave pública. De esta manera, la mera posesión de la clave privada define la identidad. Si varios nodos estuvieran en posesión de una misma clave privada, podría considerarse que se trata de una identidad distribuida.

IDENTIFICADORES DE HOST

La Identidad de Host añade dos características principales a los protocolos de Internet. La primera es el desacoplamiento de los niveles de red y transporte, que permitirá una evolución independiente de ambos niveles, además de proporcionar servicios punto a punto entre niveles de red de diferentes. La segunda característica es la autenticación de los extremos. Debido a que el HI es una clave pública, ésta puede usarse para la autenticación en protocolos de seguridad como IPsec.

En teoría, cualquier nombre de ámbito global estadísticamente único podría servir como Identificador de Host. Sin embargo, en opinión de los diseñadores de HIP, la parte pública de una pareja de claves asimétricas es la mejor opción para este cometido. La idoneidad de desacoplar los niveles de red y transporte es un tema controvertido. Después de todo, este acoplamiento ha sido durante años una garantía de seguridad y con su eliminación surge la posibilidad de nuevos tipos de ataques que podrían manipular las relaciones entre identificadores de red y transporte. HIP intenta resolver este problema a través de la verificación de clave pública.

Usando su clave privada, el emisor firma los paquetes del protocolo HIP, de modo que el receptor puede verificar su autenticidad usando el propio identificador del emisor, es decir, su clave pública correspondiente. Esto permite desarrollar mecanismos para proteger a los nodo HIP frente a ataques de intermediación (*Man in the Middle,* MitM) y denegación de servicio (*Denial of Service,* DoS). Por este motivo, sólo han se han llevado a la práctica sistemas con HI de clave pública y mensajes HIP autenticados. En las RFC de HIP se desarrolla una forma no criptográfica de HI y HIP para completar su estudio teórico, sin embargo, su uso está absolutamente desaconsejado ya que podría dar lugar a ataques aún peores que los ya conocidos sin el uso de Identidades de Host.

En el momento de elaboración de esta memoria, la RFC 5201 de HIP (17), establece que sus implementaciones tienen que soportar obligatoriamente el algoritmo de clave pública RSA/SHA1 (18), que es recomendable que soporten el algoritmo DSA (19), y que opcionalmente podrán soportar otros algoritmos. Debido a que existen diferentes algoritmos criptográficos de clave pública, que a su vez pueden usar distintas longitudes de clave, el HI no es una buena opción para su uso como identificador de paquete, o como índice en las diferentes tablas operativas necesarias para el soporte de HIP. Consecuentemente, un hash de longitud fija generado a partir del HI será la representación operativa en su lugar.

Los HI (la clave pública) pueden almacenarse en directorios como DNS o LDAP, y también se pueden enviar mutuamente entre los nodos comunicantes durante el intercambio básico, pero en ningún caso un HI se utilizará directamente en ninguno de los protocolos de Internet. La **Etiqueta de Identidad de Host** (***Host Identity Tag,*** **HIT**) es un hash de 128 bits que se usa como representación de la Identidad de Host en los paquetes del protocolo HIP y como índice en la tablas de estado de cada nodo final. Para la compatibilidad de HIP con los protocolos y APIs de IPv4, existe otra posible representación de la Identidad de Host, el **Identificador de Ámbito Local** (***Local Scope Identifier,*** **LSI**), un hash de 32 bits de longitud.

HOST IDENTITY TAG (HIT)

Un HIT es la representación de un HI en un patrón de 128 bits. Se crea realizando un hash criptográfico sobre el correspondiente HI, y su uso en los protocolos presenta un formato independiente del algoritmo de clave pública utilizado. Además, al tener una longitud fija hace más fácil la codificación del

protocolo y mejora el coste por tamaño de paquete (*overhead*) de esta tecnología.

Un HIT, tal y como está definido, tiene tres propiedades fundamentales:

- Tiene la misma longitud que una dirección IPv6 y puede utilizarse en las APIs y protocolos dentro de los campos previstos para ellas.
- Dado un HIT resulta computacionalmente difícil encontrar un HI distinto del original que encaje con él, por lo que no se considera necesaria una autoridad de certificación que verifique la relación entre HIT y HI (*self-certified*).
- La posibilidad de una colisión de HIT entre dos HIs es muy baja.

En la RFC 4843 (20) se exponen las razones para la asignación de un prefijo en el bloque de direcciones IPv6, que aloje un tipo de identificadores especiales llamados **ORCHID**[15]. Estos se definen como un hash criptográfico de 128 bits, no enrutable a nivel de red, pero enrutable en algún otro nivel superpuesto (*overlay layer*). Un HIT es un tipo concreto de ORCHID basado en el algoritmo hash SHA1 y aplicado a un Identificador de Host. Debe generarse, por tanto, siguiendo el método expuesto en la citada RFC.

El algoritmo de generación, que tomando como entrada una cadena de bits y un identificador de contexto produce como salida un ORCHID es, básicamente, el siguiente:

Entrada	**:=**	**cualquier cadena de bits**
Entrada_Hash	**:=**	**Contexto | Entrada**
Hash	**:=**	**Función_Hash(Entrada_Hash)**
ORCHID	**:=**	**Prefijo | Encode_100(Hash)**

Donde tenemos:

- **Entrada**: es una cadena de bits única o estadísticamente única que, dentro de un contexto, estará asociada con el ORCHID generado. En el caso de HIP esta cadena será el HI, es decir, la clave pública codificada en el formato de almacenamiento de DNSSEC. Los detalles de esta

[15] *Overlay Routable Cryptographic Hash Identifier.*

codificación pueden encontrarse en la RFC 3110 para claves RSA (18), o en la RFC 2536 para claves DSA (19).

- **Contexto**: es una cadena de 128 bits elegida aleatoriamente, que define el contexto de utilización previsto para el ORCHID y la función hash que deberá utilizarse para su generación en ese contexto. Estos valores son asignados en el espacio de nombres de las Etiquetas de Clase (*Type Tags*) para CGA[16], descrito en la RFC 3972 (21). En el caso de HIP, el valor asignado es:

0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA

Otros contextos asignados son, por ejemplo, para el protocolo SEND[17] definido en la RFC 3971, o para una extensión de MIP6 definida en la RFC 4866.

- **Función_Hash**: es una función hash unidireccional correspondiente al contexto identificado por el parámetro anterior. En el caso de HIP la función es SHA1, definida en la RFC 3174 (22).
- **Encode_100**: es una función en la cual la salida se obtiene extrayendo los 100 bits centrales de la cadena dada como argumento.
- **Prefijo**: una cadena constante de 28 bits, que se usa como prefijo para todas las direcciones de tipo ORCHID. En marzo de 2007, la IANA[18] asignó para este cometido el siguiente prefijo:

2001:10::/28

Este prefijo está alojado dentro del bloque de direcciones globales *unicast* de IPv6, por lo que será indistinguible de las demás direcciones del mismo tipo. Sin embargo, debe advertirse que los ORCHID, y por tanto los HIT, no cumplen los requisitos de la RFC 4291 (23) respecto a los 64 bits del identificador de interfaz, al no estar estos generados a partir de la dirección de nivel de enlace.

LOCAL SCOPE IDENTIFIER (LSI)

El Identificador de Ámbito Local (LSI) es la representación local de una HI en un patrón de 32 bits. El propósito del LSI es el de facilitar el uso de

[16] *Cryptographically Generated Addresses.*
[17] *Secure Neighbor Discovery.*
[18] *Internet Assigned Numbers Authority.*

Identidades de Host en los protocolos y APIs actuales. Las aplicaciones IPv4 pueden utilizar los LSI en lugar de direcciones IP, aunque éstos sólo tengan significado dentro del propio nodo, ya que el nivel HIP los sustituye por los HIT correspondientes para realizar todos los intercambios del protocolo. Algunos ejemplos sobre el uso de los LSI podrían ser: como dirección en un comando FTP y como dirección en una llamada a un *socket*. La ventaja de los LSI sobre los HIT está, por tanto, en su tamaño y su desventaja en la localidad de su ámbito.

Los LSI son de la forma 1.0.0.0/8, en donde los 24 bits de interfaz se obtienen a partir los correspondientes últimos bits del HIT asociado a la HI. Este rango está incluido dentro del bloque de direcciones reservadas por el IANA.

ALMACENAMIENTO DE LOS HIT EN EL DNS

El sistema DNS puede usarse para obtener un HI a partir del FQDN[19], para obtener el HIT a partir de un HI, o para verificar un HIT por medio de una búsqueda inversa. Sin embargo, el DNS no resulta apropiado para obtener la dirección IP correspondiente a la localización actual de un nodo HIP, ya que no fue diseñado para manejar cambios frecuentes y presenta problemas con el tiempo de convergencia de dichos cambios (ver, Mecanismo de punto de encuentro, servidor *randezvous*).

En la reciente RFC 5205 (24), se especifica un nuevo RR (*Resource Record*) para DNS y se describe su utilización en combinación con el protocolo HIP. Mediante dicha extensión, un nodo puede publicar su HI, su HIT, y los nombres DNS de sus servidores *randezvous*. Los identificadores anónimos, por supuesto, no deben ser almacenados en ningún sitio, aparte de los propios nodos involucrados. En el caso de servidores DNS que no incluyan las extensiones anteriores, los HIT pueden almacenarse como entradas del tipo AAAA. Sin embargo esto puede resultar una potencial fuente de problemas dado que las aplicaciones pueden, inadvertidamente, tomar los HIT por direcciones IPv6 enrutables.

De forma alternativa, además de almacenar los HI en DNS, pueden almacenarse en otros tipos de Infraestructuras de Clave Pública (PKI). Esta práctica permitiría que fueran utilizadas para otros usos distintos de la identificación de nodos.

[19] *Full Qualified Domain Name*.

UNA NUEVA ARQUITECTURA DE LA PILA IP

En la arquitectura HIP se incorpora un nuevo nivel, el de host, de modo que los nombres de extremos y localizadores están diferenciados entre sí. Las direcciones IP siguen actuando como localizadores, mientras los HI asumen el papel de identificadores de extremo. Una visión del cambio de relaciones entre las entidades lógicas y la pila de protocolos se muestra en el siguiente gráfico.

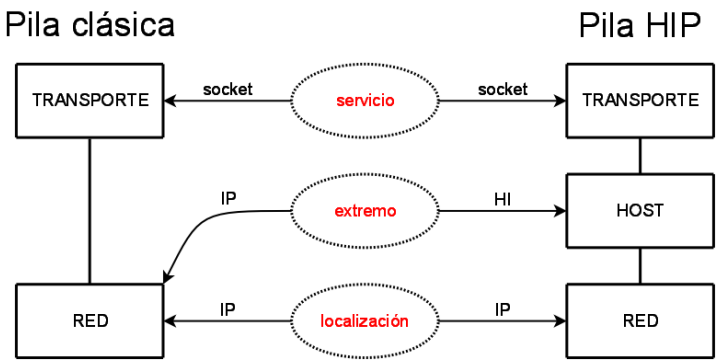

Ilustración 1: relaciones entre las entidades lógicas y la pila de protocolos.

EXTREMOS Y ASOCIACIONES DE TRANSPORTE

Arquitecturalmente hablando, HIP estipula ligaduras diferentes para los protocolos de nivel de transporte. Esto es, las asociaciones de nivel de transporte, o sea, las conexiones TCP, las asociaciones UDP y las asociaciones de seguridad IPsec, ya no están ligadas a direcciones IP sino a Identidades de Host, como se muestra en el siguiente gráfico.

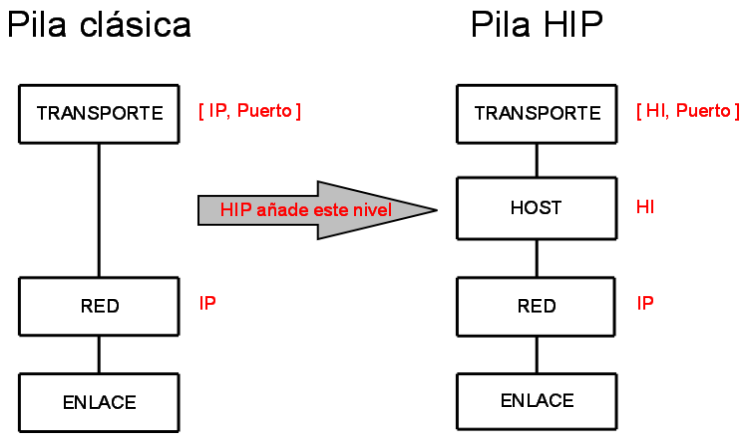

Ilustración 2: niveles de la pila y sus identificadores.

Es posible que un único sistema físico dé alojamiento a varios extremos lógicos. Con HIP, cada uno de esos extremos tendrá una Identidad de Host diferente. Debido a que las asociaciones de nivel de transporte están ligadas a Identidades de Host, HIP puede afrontar mejor la migración de procesos y los servicios en *cluster*. Es decir, si una Identidad de Host se mueve de un sistema físico a otro, es posible mover con ella todas sus asociaciones de transporte sin romperlas. De forma similar, si fuera posible distribuir el procesamiento de una única Identidad de Host entre varios sistemas físicos, HIP podría proporcionar servicios en *cluster* sin hacer ningún cambio en el extremo del cliente.

MOVILIDAD EXTREMO A EXTREMO Y *MULTIHOMING*

HIP desacopla los niveles de red y transporte, y fija las asociaciones de transporte con Identidades de Host (aunque realmente sean HIT o LSI). Por consiguiente, HIP puede proporcionar movilidad y *multihoming* a nivel de red con un coste muy bajo en términos de infraestructura. Un sistema se considera móvil si su dirección IP puede cambiar dinámicamente por cualquier motivo, ya sea DHCP, autoconfiguración IPv6, establecimiento de un enlace PPP o actualización de un dispositivo NAT. Igualmente, un sistema se considera *multihomed* si tiene al mismo tiempo más de una dirección IP enrutable de ámbito global. HIP trabaja agrupando todas las direcciones IP cuando estas pertenecen a la misma Identidad de Host. En el caso de que alguna dirección quede inutilizable, u otra con mayor preferencia aparezca disponible, las asociaciones de transporte preexistentes pueden migrarse a otra dirección con facilidad.

Cuando un nodo cambia su IP mientras hay en marcha una comunicación activa, es decir, existen conexiones de transporte abiertas con otros nodos que llamaremos corresponsales, los cambios de dirección se gestionan en modo directo. El corresponsal del nodo móvil sólo tiene que aceptar los paquetes HIP debidamente autenticados y los paquetes IPsec con protección de integridad, independientemente de la dirección IP de la que provengan. Por otro lado, como se describirá más adelante, el nodo móvil debe enviar un paquete de re-direccionamiento HIP para informar a sus corresponsales de su nueva dirección o direcciones. A su vez, los nodos corresponsales deberán verificar que el nodo móvil es alcanzable a través de dichas direcciones. Como ya se verá en la sección de protección contra ataques DoS, esto es especialmente importante en aquellas situaciones en que el nodo corresponsal envía datos periódicamente al nodo móvil, reanudando la comunicación por una asociación establecida previamente.

INTERCAMBIO BÁSICO Y ASOCIACIONES HIP

El intercambio básico HIP (*Basic Exchange*, BEX), definido en la RFC 5201 (17), es un protocolo criptográfico con dos participantes que se utiliza para gestionar el establecimiento de un contexto de comunicaciones entre extremos, denominado asociación HIP. Por definición, el sistema que arranca un intercambio BEX es el **iniciador** (***initiator***), y su corresponsal es el **contestador** (***responder***). Esta distinción deja de tenerse en cuenta una vez que el intercambio básico se ha completado, y cualquiera de las partes puede actuar como iniciador en comunicaciones futuras. Una asociación HIP establecida puede actualizarse utilizando el mecanismo de actualización definido, y cuando dicha asociación deja de ser necesaria, puede cerrarse usando el apropiado mecanismo de cierre. A continuación veremos todos estos elementos del protocolo de forma detallada.

El primer paquete, I1, inicia el BEX y junto a los tres paquetes siguientes, R1, I2 y R2, constituyen un intercambio Diffie-Hellman autenticado que genera una clave de sesión. Este material criptográfico se utiliza para obtener las claves de la asociación HIP y cualquier otra clave criptográfica que se necesite, como por ejemplo las utilizadas por el protocolo ESP.

Primeramente, el iniciador envía un paquete disparador, **I1**, al contestador. El paquete sólo contiene el HIT del iniciador y el del contestador, si es que lo conoce. En algunos casos sería posible reemplazar este I1 por algún otro tipo de mecanismo de arranque, en cuyo caso el protocolo comienza con el envió de un R1 por parte del contestador.

El segundo paquete, **R1**, comienza el intercambio efectivo. Contiene un puzle, es decir, un desafío criptográfico que el iniciador debe resolver. El nivel de dificultad del puzle se puede ajustar según el nivel de confianza con el iniciador, la carga del sistema, o cualquier otro motivo. Además, el R1 contiene los parámetros iníciales del intercambio Diffie-Hellman y una firma que cubre parte del mensaje. Se mantienen algunos campos fuera de la firma para poder usar paquetes R1 pre-calculados.

En el tercer paquete, **I2**, el iniciador debe mostrar la solución del puzle recibido. Si la solución no es correcta el I2 será descartado. Además, el I2 contiene un parámetro Diffie-Hellman y una firma. Finalmente, el paquete **R2**, también firmado, termina el intercambio y puede comenzar a continuación el tráfico de paquetes de datos. En el futuro, los paquetes I2 y R2 también podrían llevar

carga de datos, sin embargo, los detalles acerca de esto se han retrasado hasta que se disponga de una mayor experiencia práctica.

En el siguiente esquema puede verse todo el proceso gráficamente. En él, los términos D-H son los parámetros Diffie-Hellman y los HI los Identificadores de Host que intercambian los participantes.

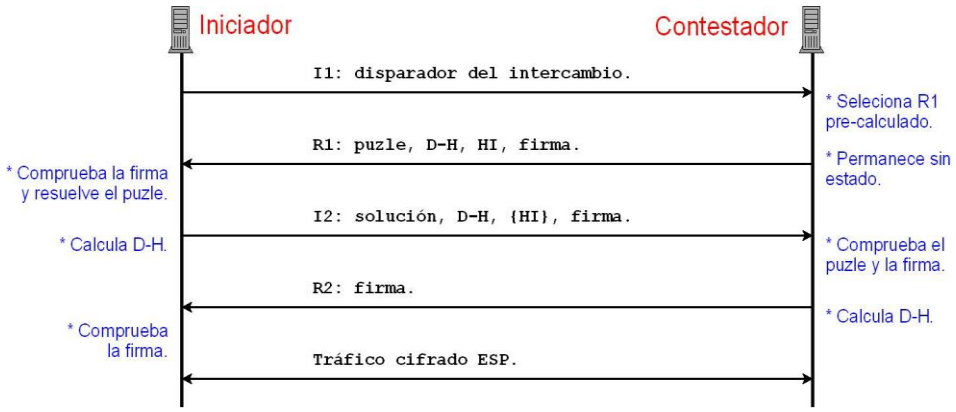

Ilustración 3: Intercambio Básico HIP.

El iniciador puede usar la salida Diffie-Hellman para cifrar su HI en el I2 (aunque algunos autores advierten que esto puede interferir en la inspección de paquetes por posibles entidades intermedias (*middleboxes*)), o bien enviarlo en claro. Por su parte, el HI del contestador no está protegido. Debe advertirse, sin embargo, que los HIT tanto del iniciador como del contestador se ponen en claro en los paquetes, permitiendo a un oyente furtivo saber a priori que intervinientes están verificando sus identidades.

Como mecanismo de cierre de una asociación HIP se definen dos nuevos paquetes firmados, denominados **CLOSE** y **CLOSE_ACK**, que se intercambiarán para solicitar y confirmar el cierre respectivamente. Por último, existe un paquete opcional de notificación, denominado **NOTIFY**, que permite a un nodo informar a su corresponsal sobre determinadas eventualidades, como errores del protocolo o fallos de negociación.

En la sección de apéndices de este trabajo puede consultarse la máquina de estados completa del intercambio básico.

MECANISMO DE PUZLE

El propósito del mecanismo de puzle es el de evitar determinados ataques de negación de servicio (DoS). Este mecanismo permite al contestador retrasar la creación de un estado de conexión hasta la recepción del I2, pero además, también sirve para comprobar de forma económica en términos computacionales la sinceridad del iniciador, demostrada por los ciclos de CPU gastados en resolver el puzle.

Este mecanismo se ha diseñado explícitamente para dar cabida a varias opciones de implementación. Por un lado, permite una implementación del contestador que aplace completamente la creación de un estado de conexión hasta la recepción de un I2 valido. En tal caso, un paquete I2 con formato correcto sólo puede ser rechazado por el contestador después de comprobar su validez evaluando una función *hash*. Pero, por otro lado, el diseño también permite implementaciones del contestador que conserven un estado para los paquetes I1 recibidos, de modo que pueda filtrar los I2 posteriores evitando, en su caso, el coste computacional de dicha función *hash*. El inconveniente de este último enfoque es la necesidad de crear dicho estado. Por supuesto, también se permiten implementaciones con mecanismos combinados de ahorro de memoria y tiempo de computación.

Una manera de que el contestador permanezca sin estado y al mismo tiempo sea capaz de filtrar la mayoría de I2 falsificados, consiste en basar la selección del puzle en alguna función sobre la identidad del iniciador. La idea fundamental es que el contestador disponga de un cierto número de paquetes R1 pre-calculados, eligiendo uno de ellos en función de la información del iniciador recibida en el I1. Más tarde, cuando el contestador recibe un I2, éste puede comprobar que el puzle enviado en el R1 coincide con el recibido. Así, resulta impracticable para un atacante la táctica de intercambiar un I1/R1 y, a continuación, generar un gran número de I2 falsificados que aparentemente vienen de distintas direcciones IP o usan diferentes HIT. Este método, sin embargo, no protege contra el mismo tipo de ataque si se utiliza una IP y un HIT fijos. Una posible solución contra dicho ataque consistiría en la creación de un pseudo-estado que permita recordar verificaciones de puzle fallidas y relacionarlas con su IP y HIT de origen.

El contestador puede establecer la dificultad del puzle basándose en su nivel de confianza con el iniciador. Como el puzle no está incluido en el cálculo de la firma, el contestador puede utilizar R1 pre-calculados, incluyendo el puzle en el último momento antes de enviar el paquete al iniciador. Además, el

contestador puede usar técnicas heurísticas para determinar cuándo se encuentra bajo un ataque DoS, ajustando en consecuencia los parámetros de dificultad del puzle.

INTERCAMBIO DEL PUZLE

El contestador comienza el intercambio del puzle cuando éste recibe un I1. Entonces, el contestador proporciona al iniciador un número aleatorio I, solicitándole que encuentre un determinado número J. Para obtener el valor de J, el iniciador debe aplicar un cierto algoritmo *hash* sobre la concatenación de I, J y los HIT de ambas partes, de modo que los K bits de menor orden del resultado sean ceros. El valor de K establece, por tanto, la dificultad del puzle.

Para encontrar el valor adecuado de J, el iniciador tiene que generar y probar una serie de valores hasta que uno de ellos satisfaga la condición sobre el número de ceros. El puzle tiene un tiempo de vida dado por el contestador, transcurrido el cual dejará de ser válido. El contestador, por su parte, puede verificar que el iniciador ha completado la tarea asignada calculando el *hash* una sola vez.

Para evitar ataques con respuestas pre-calculadas, el contestador debe seleccionar el número I de manera que el iniciador no pueda averiguarlo a priori. Además, el mecanismo debe permitir al contestador la posibilidad de verificar que dicho número ha sido efectivamente elegido por él y no por el iniciador. Para esto, usando un campo de datos opaco (parámetros ECHO_REQUEST_SIGNED o ECHO_REQUEST_UNSIGNED) el contestador puede incluir un dato en el paquete R1 que el iniciador deberá copiar intacto en el correspondiente I2. Este dato opaco se puede generar de varias formas, por ejemplo, mediante un *hash* de la concatenación del propio número I, algún dato secreto generado localmente y, posiblemente, otros datos relacionados como la IP o el HIT del iniciador. De esta forma, usando ese mismo secreto, la I recibida en el paquete I2 y los otros datos necesarios, el contestador puede verificar que la I recibida es la misma que la enviada. El dato secreto deberá cambiarse frecuentemente.

Es recomendable que el contestador genere un nuevo puzle y nuevo paquete R1 pre-calculado cada pocos minutos. Además, también puede ser recomendable que conserve los puzles descartados durante, al menos, el equivalente al doble de su tiempo de vida. Ambas medidas permiten trabajar

con un iniciador lento, y a la vez limitar la utilidad que tiene un puzle ya resuelto para un atacante.

El contestador dispone de un **contador de generaciones de R1** (R1 *generation counter*) que irá incrementándose conforme se renuevan el puzle y el R1. Este contador puede incluyese dentro del paquete R1 y, entre otras cosas, indica al iniciador cuál es la generación del puzle recibido (ver el apartado de protección contra reenvíos). Un iniciador debe aceptar puzles de la generación actual, pero no está obligado a aceptar los de generaciones anteriores.

PROTOCOLO DIFFIE-HELLMAN AUTENTICADO

Los paquetes R1, I2 y R2 conforman un intercambio Diffie-Hellman autenticado estándar. El contestador, en su R1, envía uno o dos valores Diffie-Hellman públicos y su clave pública de autenticación, esto es, su Identificador de Host. La firma del R1 permite al iniciador verificar que el paquete ha sido enviado por el contestador. Sin embargo, puesto que es un paquete pre-calculado y la firma no lo cubre en su totalidad, esto no es bastante para evitar ataques de reenvío (*replay*) que se tratarán más adelante.

Cuando el iniciador recibe un R1, obtiene uno o dos valores Diffie-Hellman públicos provenientes del contestador. Selecciona el correspondiente al del grupo más fuerte que pueda soportar, calcula la clave de sesión y crea una asociación HIP. A continuación, envía un paquete I2 que contiene su valor Diffie-Hellman público y su clave pública de autenticación, opcionalmente cifrada. La firma cubre la totalidad del paquete I2.

Finalmente, el contestador extrae del I2 el valor público Diffie-Hellman del iniciador, calcula la clave de sesión y crea la correspondiente asociación HIP. Después, descifra la clave pública de autenticación del iniciador y verifica la firma del paquete usando dicha clave.

El último paquete, R2, se envía únicamente para proteger al iniciador contra ataques de reenvío de paquetes R1. Al recibir un R2, el iniciador pasa la conexión al estado ESTABLECIDO, en el cual descartará directamente cualquier R1 que, verificando su HMAC[20], se delatara como reenvío.

[20] *Hash Message Authentication Code*. Es un tipo de código de autenticación calculado mediante un algoritmo hash en combinación con una clave secreta. Es computacionalmente más ligero que la verificación de una firma.

PROTECCIÓN CONTRA REENVÍOS

El protocolo HIP contiene los siguientes mecanismos de protección contra reenvíos malintencionados. El contestador está protegido contra reenvíos de paquetes I1 gracias al sistema de respuesta sin estado mediante paquetes R1 pre-calculados. El iniciador está protegido contra reenvíos de paquetes R1 mediante el denominado contador de generaciones de R1 (R1 *generation counter*) incluido en el propio paquete R1. El contestador está protegido contra reenvíos o falsificaciones de paquetes I2 por el mecanismo de puzle y opcionalmente el uso datos opacos. Finalmente, un nodo está protegido contra reenvíos de paquetes R2 y paquetes UPDATE (ver, actualización de asociaciones HIP) por medio de una verificación HMAC, menos pesada y previa a la comprobación de la firma.

El contador de generaciones de R1 es una implementación en 64 bits de una función monótona creciente que puede iniciarse en cualquier valor. El nodo puede tener un único contador de ámbito común para todo el sistema o uno por cada Identificador de Host, en el caso de que haya más de uno. El valor de este contador debe ser persistente, es decir, conservarse después del reinicio de los procesos HIP e incluso del propio hardware. Debe incrementarse, al menos, cada vez que se descarta un R1 antiguo (ver el capítulo sobre el mecanismo de puzle), y no debe decrementarse ni darse la vuelta en ningún caso, so pena de poner a sus corresponsales en riesgo de ataque por reenvío.

Un nodo podría recibir más de un R1, debido al envío de varios I1 o bien al reenvío de un R1 antiguo. Cuando se envían múltiples I1s, el iniciador deberá esperar un tiempo razonable tras la recepción del primer R1 (por ejemplo, el doble del *roundtrip* estimado), de modo que se permita la llegada del resto, y a continuación responder sólo a uno de entre los que tengan el valor más alto del contador de generaciones de R1. Si mientras el iniciador está procesando el R1, o incluso mientras se encuentra a la espera de recibir el R2, le llegase un R1 con un valor mayor de dicho contador, tendrá la opción de reiniciar el proceso con el nuevo R1 recibido, como si éste fuera el primero.

RECHAZO DEL INTERCAMBIO

Un nodo tiene la potestad de elegir si acepta o no un intercambio HIP. Si la política del nodo es la de actuar sólo como iniciador, éste puede responder a los I1 iniciando su propio intercambio. El motivo para que un nodo se comporte

así está en que el HI del iniciador es el único protegido durante el intercambio. Sin embargo, si ambos nodos intentan dicha política el intercambio fallará y existe el riesgo de que se produzca una condición de carrera.

Si la política de un nodo no permite entrar en un intercambio HIP con un iniciador, éste deberá enviar un mensaje ICMP del tipo "Destino Inalcanzable, Prohibido por el Administrador" ("*Destination Unreachable, Administratively Prohibited*"). Los diseñadores del protocolo HIP han descartado el uso en estos casos de un paquete HIP más complejo, ya que abriría la posibilidad de otros ataques DoS.

MODO OPORTUNISTA

Resulta posible iniciar un intercambio básico HIP incluso desconociendo el HI y el HIT del contestador. Para ello, el paquete iniciador I1 contendrá el HIT nulo (todo ceros) en el lugar destinado al HIT del contestador. Esta clase de establecimiento de conexión se conoce como "modo oportunista", y plantea algunos problemas relacionados con las APIs y la seguridad.

Dado que el iniciador no conoce el HI del contestador, deberán existir en el API las llamadas adecuadas que permitan al *kernel* subyacente iniciar el intercambio básico HIP basándose únicamente en los localizadores. El HI del contestador estará disponible en el R1, con la debida reserva y de una forma ya autenticada, cuando el R2 haya sido recibido y verificado. De esa forma, éste podrá pasarse al nivel aplicación mediante una llamada del API diseñada al efecto. Sin embargo con un API compatible hacia atrás, la aplicación sólo podrá ver los localizadores usados para el intercambio inicial. Actualmente, éste es un área de estudio activa; por ejemplo, se trabaja en la creación de un API que pueda usarse por las aplicaciones para especificar sus requerimientos de seguridad en este contexto (25).

Además de esto, resultan pertinentes las siguientes consideraciones de seguridad. El mecanismo contador de generaciones de R1 resulta menos eficiente en la protección contra reenvíos de paquetes R1, ya que el contestador puede reenviar cualquier R1, independientemente de su HI, en lugar de verse obligado por un HI explicito en el I1. Y aún más importante, un intercambio en modo oportunista es vulnerable a ataques de intermediación (MitM – *Man In The Middle*), debido a que el iniciador no dispone de ninguna información sobre la clave pública del contestador. Para valorar el impacto de

esta vulnerabilidad la compararemos con las vulnerabilidades actuales de las comunicaciones no HIP.

Un atacante que se encuentre en el camino entre dos nodos puede ponerse a sí mismo como intermediario, proporcionando su identidad al iniciador y estableciendo una nueva asociación HIP con el contestador. Un atacante fuera del camino entre los nodos será incapaz de hacer tal cosa, ya que no puede interceptar y suplantar los mensajes del intercambio básico. Para que esto sea posible sin que los comunicantes lo adviertan, el iniciador debe usar el modo oportunista y el contestador debe estar configurado para aceptar conexiones de cualquier nodo HIP.

Estas puntualizaciones también son aplicables a las comunicaciones a través de la Internet actual. Un cliente que contacte con un servidor sin usar seguridad punto a punto podría encontrarse inadvertidamnete hablando con él a través de un intermediario, asumiendo de nuevo que dicho servidor esté dispuesto a hablar con todo el mundo.

Si se usa seguridad punto a punto, entonces lo peor que puede ocurrir, tanto con HIP en modo oportunista como con IP, será un caso de denegación de servicio. Una entidad en el camino podrá interrumpir la comunicación, pero será incapaz de colocarse como intermediario. Sin embargo, una vez realizado correctamente el intercambio básico, HIP proporciona integridad y confidencialidad de comunicaciones permitiendo, además, cambiar de forma segura los localizadores de los extremos. Como conclusión podríamos decir que el protocolo HIP en modo oportunista es, al menos, tan seguro como el actual protocolo IP.

POLÍTICAS HIP

Existen ciertas variables que influyen en los intercambios HIP que cada nodo debe soportar. Todas las implementaciones de HIP deben soportar al menos dos HIs, uno para publicar en algún tipo de directorio (p.e. DNS) y otro inédito para su uso como identificador anónimo. Aunque los HIs anónimos rara vez se utilizarán como contestador, probablemente serán iniciadores de uso común. Se recomienda, por tanto, el soporte de múltiples HI.

Es probable, que la mayoría de iniciadores quieran usar diferentes HI para diferentes contestadores. Las implementaciones deben proporcionar una política de HIT iniciador a HIT contestador. Estas políticas deberán incluir

también transformaciones preferentes y tiempos de vida locales. Los contestadores también necesitan una política similar, describiendo los nodos autorizados a participar en determinados intercambios HIP, sus transformaciones preferentes y sus tiempos de vida locales.

Por último cabe reseñar, que el protocolo HIP está diseñado como un protocolo de autenticación e intercambio de claves extremo a extremo, para ser usado con ESP u otros protocolos de seguridad extremo a extremo. Sin embargo, éste no cubre todo el control de políticas detalladas de que dispone IKE, y que le permiten afrontar la definición de políticas complejas necesarias, por ejemplo, en determinas pasarelas (*gateways*). Por lo tanto, HIP no es, ni pretende ser, un sustituto de IKE.

ESP COMO TRANSPORTE DE DATOS EN HIP

El intercambio básico HIP proporciona un medio para realizar la autenticación mutua de los extremos, generar material criptográfico y establecer una asociación de seguridad, pero lo que no proporciona es un medio de transporte seguro para los datos de usuario. En la RFC 5202 (26) se especifican las extensiones que permiten el uso del protocolo ESP para este cometido, siendo, a día de hoy, el único método completamente definido y de inclusión obligatoria para todas las implementaciones de HIP. En el futuro, podrán desarrollarse otros métodos para el transporte de los datos, incluyendo algunos que no hagan uso de protección criptográfica.

Llevar el HI y el HIT en todas las cabeceras de paquetes de datos de usuario incrementaría notablemente la sobrecarga, por lo tanto, deberán usarse otros métodos para asociar cada paquete de datos con su correspondiente HI. Utilizando ESP para proteger el tráfico de datos, es posible usar HIP sin añadir ninguna cabecera en los paquetes de usuario. El **SPI** (***Security Parameter Index***) incluido en la cabecera de ESP puede servir para relacionar cada paquete cifrado con la asociación HIP apropiada.

El protocolo ESP (27) es un protocolo de transporte, definido dentro del marco de IPsec (7), que siempre debe trabajar en conjunto con protocolos criptográficos, como puede ser IKEv2 (10), para el establecimiento y la gestión de las asociaciones de seguridad (SA). En el caso de HIP, el intercambio básico hace las funciones de un protocolo criptográfico ligero, adaptado al establecimiento de asociaciones de seguridad ESP con una menor granularidad de políticas, es decir, con una semántica limitada al modo transporte extremo a extremo (ver, ESP modo BEET).

Aunque sería posible, al menos en teoría, utilizar algún otro protocolo criptográfico actual con Identificadores de Host para establecer las SA, HIP prefiere definir un nuevo protocolo. Para esto existen un cierto número de razones históricas, y también algunas razones de arquitectura. Por ejemplo, IKEv2 no se diseño teniendo en cuenta entidades intermedias. Al añadir un nuevo nivel de nombres, se añade la posibilidad de un nuevo nivel de reenvío (*forwarding*), por lo que es importante un diseño del protocolo HIP que facilite el trabajo de dichas entidades intermedias.

ESTABLECIMIENTO Y GESTIÓN DE LAS SA

El establecimiento de asociaciones de seguridad ESP entre dos nodos HIP consiste en tres mensajes incorporados mediante parámetros a los paquetes R1, I2 y R2 del intercambio básico.

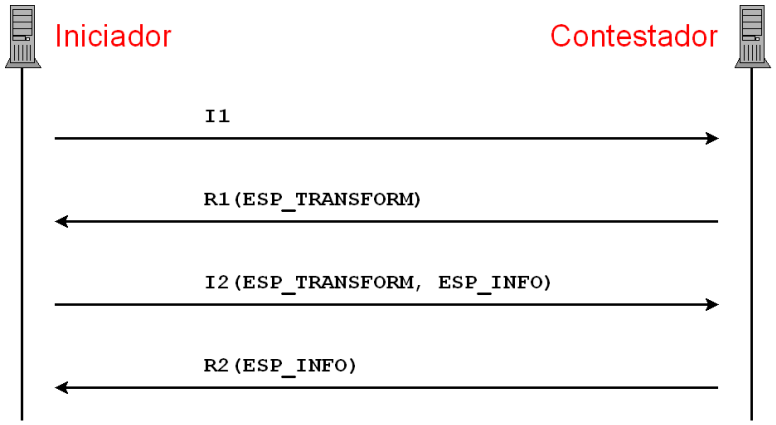

Ilustración 4: establecimiento de asociaciones ESP durante el BEX.

En el mensaje R1 se incluye el parámetro ESP_TRANSFORM, en el cual el contestador declara todas las transformaciones ESP que está dispuesto a usar. A su recepción, el iniciador debe seleccionar una de ellas e incluirla en el paquete I2 mediante otro parámetro ESP_TRANSFORM. Además, incluirá un parámetro ESP_INFO conteniendo el valor de SPI asignado a los paquetes que recibirá de su corresponsal. Por último, el paquete R2 incluirá otro parámetro ESP_INFO con el valor de SPI seleccionado por el iniciador. Completado el intercambio anterior, quedan establecidas dos asociaciones de seguridad ESP que se agrupan en una asociación HIP. Sin embargo, a lo largo de la vida de una asociación HIP, puede ser necesario actualizar, o incluso cambiar, la asociaciones ESP.

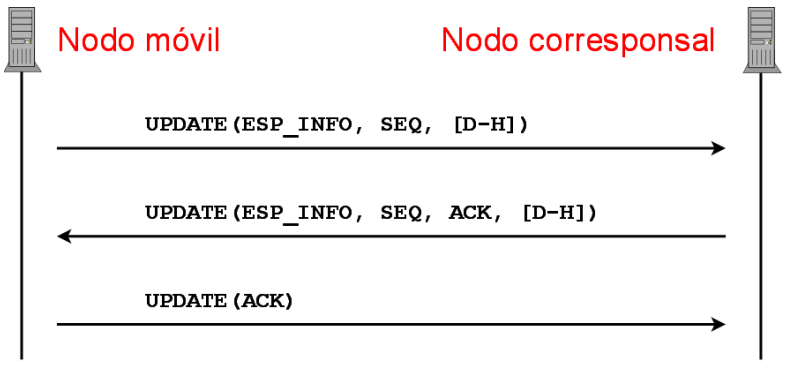

Ilustración 5: intercambio para actualización de asociaciones ESP.

Para la actualización sólo se utilizan dos mensajes que se envían mediante paquetes UPDATE (ver, Actualización de una asociación HIP). Cuando un nodo HIP desee actualizar una asociación ESP incluirá un parámetro ESP_INFO conteniendo el valor del SPI actual, el nuevo SPI y el índice criptográfico, necesario para obtener la siguiente clave. Si fuera necesario generar nuevo material criptográfico se añadiría un parámetro DIFFIE-HELLMAN, ya definido para el intercambio básico.

PROCESAMIENTO DE ESP EN MODO BEET

La especificación actual de IPsec define dos únicos modos de operación: modo túnel y modo transporte. El **modo túnel** está pensado principalmente para usos no punto a punto, donde uno o ambos extremos de las asociaciones de seguridad (SA) están ubicados en pasarelas de seguridad (*gateways*), distintas de los nodos finales. El **modo transporte** está pensado para un uso punto a punto, donde ambos extremos de las asociaciones de seguridad se encuentran en los propios nodos finales. Sin embargo, este segundo modo se utiliza escasamente debido a varias razones, entre las que se incluye la mutabilidad de las direcciones IP. Es decir, debido al uso de NAT, la movilidad o el *multihoming*, las direcciones que realmente circulan por la red no coinciden necesariamente con las que las aplicaciones esperan ver. Atravesar un NAT es actualmente un importante problema para IPsec. No es suficiente con encapsular los paquetes en UDP; además, el uso del modo túnel es imprescindible, ya que la dirección IP externa al final de la conexión protegida es diferente. Si se usara el modo transporte, la diferencia en las direcciones IP provocaría el fallo de los *checksum* TCP/UDP.

Basándonos en las razones anteriores, encontramos la necesidad de un modo transporte en el que las direcciones que ven las aplicaciones sean distintas de las que realmente se ponen en la red. El modo túnel, actualmente, proporciona dicha funcionalidad, pero al coste de una sobrecarga adicional en términos de procesamiento y tamaño de paquete. El nuevo **modo BEET** (*Bound End to End Tunnel*) para ESP tiene como objetivo proporcionar una versión limitada de la semántica del modo túnel, pero sin incurrir en la sobrecarga asociada al modo túnel tradicional. Como su propio nombre indica, el modo BEET está pensado únicamente para su uso punto a punto. Proporciona una semántica de modo túnel en el sentido de que la dirección IP vista por la aplicación y la usada en la red son diferentes, creando el efecto de que la dirección IP usada por el nivel de aplicación está tunelizada sobre la dirección IP del nivel de red. Sin embargo, este modo no proporciona una semántica completa del modo túnel. Más

concretamente, la dirección IP vista por la aplicación está estrictamente fijada, y sólo puede usarse un único par de direcciones internas fijas en cada SA. En esto se diferencia del modo túnel normal, donde las direcciones IP internas pueden ser cualesquiera dentro de un rango especificado.

Las SA en modo BEET almacenan dos pares de direcciones, llamadas direcciones internas (*inner*) y direcciones externas (*outer*). Las direcciones externas son las que circulan por la red. Y dado que las direcciones internas son fijas para toda la vida de la asociación de seguridad, no es necesario enviarlas en cada uno de los paquetes. Por el contrario, estas quedan establecidas cuando se crea la asociación de seguridad, se verifican cuando los paquetes van a ser enviados, y se restauran cuando los paquetes han sido recibidos. Todo esto confiere al modo BEET la eficiencia del modo transporte y la semántica de un túnel extremo a extremo. Aunque la semántica del modo BEET está limitada en el sentido de que sólo se permite un par fijo de direcciones internas, las direcciones externas pueden variar a lo largo del ciclo de vida de la asociación de seguridad. En caso de ser necesario un nuevo par de direcciones internas, en modo BEET se deberá establecer un nuevo par de asociaciones de seguridad.

En el *draft* de Nikander (28) se realiza un estudio pormenorizado de los beneficios del modo BEET en diferentes escenarios, como son, traspasar dispositivos NAT o proporcionar seguridad para Mobile IP u otras soluciones de movilidad y *multihoming*.

En el caso de HIP, el procesamiento de ESP en modo BEET es parte integral del protocolo en su conjunto. Aunque es posible implementar HIP mediante el modo transporte tradicional, el modo BEET facilita mucho la tarea pudiendo usar HIT directamente como direcciones internas inmutables, y como externas direcciones IP variables. El SPI se utiliza en ESP para identificar la SA correcta para cada paquete recibido. Con HIP, el SPI adquiere un significado añadido: es la representación comprimida de una pareja de HIT.

Una pareja de SA vienen indexadas por sus SPI y la pareja de HIT (ambos HIT son necesarios ya que un nodo puede tener varios HIT). Las SA no necesitan fijarse a una dirección IP; todo el control interno de la SA se realiza en base a HIT. Por lo tanto, un nodo podrá cambiar su dirección IP y aun así mantener sus SA en funcionamiento. A su vez, como los protocolos de transporte están ligados a las SA, cualquier conexión de transporte activa también podrá mantenerse. Así, situaciones como la pérdida de una conexión PPP y su restablecimiento, o el tránsito de un nodo móvil, no supondrán la necesidad de nueva negociación HIP ni la interrupción del servicio de transporte.

ACTUALIZACIÓN DE UNA ASOCIACIÓN HIP

Una asociación HIP puede necesitar actualizarse a lo largo del tiempo. Ejemplos de esta necesidad son la renovación de claves de las SA para datos de usuario, añadir nuevas SA, o cambiar las direcciones IP ligadas a los nodos. El protocolo HIP proporciona un paquete UPDATE de propósito general, el cuál puede llevar múltiples parámetros para actualizar el estado de una asociación entre dos comunicantes. Los paquetes UPDATE llevan un número de secuencia monótono creciente (parámetro SEQ) y requieren un acuse de recibo explícito (parámetro ACK), el cuál devuelve números de secuencia individuales recibidos del corresponsal. Un solo paquete UPDATE puede contener un parámetro SEQ y uno o varios ACK, para acusar recibo de múltiples paquetes. Por último, los paquetes UPDATE están protegidos por un hash HMAC y una firma (parámetros HMAC y HIP_SIGNATURE). Procesar únicamente la firma, más pesada computacionalmente, puede ser causa de ataques de denegación de servicio.

En la RFC 5206 (29) se exponen los detalles sobre las extensiones del protocolo básico para la movilidad y el *multihoming*, y se definen los parámetros necesarios. Resumidamente, cuando un nodo se mueve a una nueva localización y cambia su dirección IP, lo comunicará a sus corresponsales con un paquete UPDATE que lleve uno o más parámetros LOCATOR. Las direcciones IP ligadas a una determinada asociación HIP pueden encontrarse en distintos estados. Al recibir una nueva dirección, ésta tendrá siempre el estado NO_VERIFICADA, es decir, que aún no se ha comprobado que el HI correspondiente se encuentra alcanzable en esa dirección (para esta tarea se utilizan los parámetros ECHO_REQUEST y ECHO_RESPONSE); en caso afirmativo pasará al estado ACTIVA. Una dirección ACTIVA volverá a NO_VERIFICADA si no recibe tráfico durante un tiempo estipulado por la política local del nodo. Cualquier dirección que superé su tiempo de vida (*locator lifetime*) pasará al estado DEPRECIADA. Si se volviera a recibir la misma dirección, ésta podría pasar a NO_VERIFICADA, pero nunca directamente a ACTIVA.

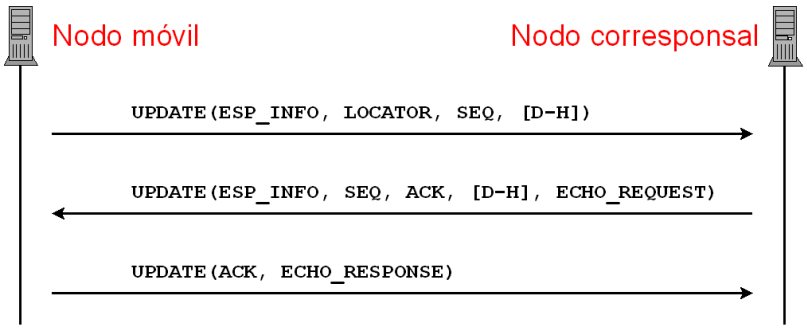

Ilustración 6: intercambio de actualización.

Si se utiliza ESP para el tráfico de datos de usuario, tras un cambio de dirección puede ser necesario renovar las claves de las asociaciones de seguridad ESP. En dicho caso se incluirán parámetros ESP_INFO con los valores de los SPI nuevo y antiguo. Opcionalmente, también se puede añadir un parámetro DIFFIE_HELLMAN para generar nuevo material criptográfico.

PROTECCIÓN CONTRA ATAQUES DOS

La idea de informar sobre los cambios de dirección IP únicamente enviando un paquete con la nueva resulta atractivamente simple, sin embargo, no es suficientemente segura. Aunque HIP no depende de la dirección IP origen para nada (una vez que el intercambio básico se ha completado), parece necesario comprobar que el nodo móvil es alcanzable en su nueva dirección IP antes de continuar enviándole tráfico de datos.

Aceptar nuevas direcciones a ciegas podría conducir a potenciales ataques DoS contra terceros. Para lograr un ataque distribuido de inundación, el atacante podía abrir una gran cantidad de conexiones HIP con un gran número de nodos usando HI anónimos y, a continuación, anunciar a todos esos nodos que se ha movido a la dirección IP de la víctima. Si los nodos corresponsales simplemente aceptaran la nueva IP, el resultado sería una avalancha de paquetes contra esa dirección. Para eliminar esta posibilidad HIP incluye un mecanismo de comprobación, el cual verifica que un nodo es alcanzable en cada dirección por separado antes de usarla para el envío de tráfico en mayores cantidades.

Cuando HIP se utiliza entre dos nodos que tienen una total confianza mutua, éstos pueden opcionalmente desactivar el mecanismo de comprobación de la dirección. Sin embargo, tal optimización de rendimiento debería limitarse a corresponsales con condiciones ideales de seguridad (acreditada confianza y capacidad de protegerse a sí mismos contra virus, troyanos u otros programas malintencionados).

MECANISMO DE PUNTO DE ENCUENTRO (SERVIDOR *RANDEZVOUS*)

Hacer el primer contacto con un nodo móvil en itinerancia resulta ligeramente más complicado con HIP que en el caso de disponer de una localización inicial fija, ya que para comenzar el intercambio HIP, el nodo iniciador tiene que saber cómo llegar hasta el nodo móvil. Los nodos móviles HIP podrían usar un DNS dinámico para actualizar su información de localización, pero el sistema DNS no está diseñado para manejar cambios muy frecuentes y presenta problemas con el tiempo de convergencia de dichos cambios.

Una alternativa al DNS consiste en utilizar un nuevo elemento estático de infraestructura para facilitar un punto de encuentro entre los nodos HIP. Un nodo móvil puede mantener continuamente informada a la infraestructura de punto de encuentro de su actual dirección o direcciones IP, confiando en ella para mantener adecuadamente las relaciones entre HIT y dirección IP.

El mecanismo de punto de encuentro es también necesario si ambos nodos coinciden en cambiar su dirección al mismo tiempo, ya sea porque son nodos móviles y se han movido a la vez, porque uno de ellos ha estado desconectado por algún tiempo, o por cualquier otra razón. En tal caso, los paquetes de re-direccionamiento se cruzarían por la red y nunca alcanzarían su destino.

En la RFC 5204 (30) se definen las extensiones del protocolo HIP para el uso de un servidor punto de encuentro o *randezvous* (RVS). Los clientes de este servidor serán nodos HIP que utilizan las extensiones de registro HIP definidas en la RFC 5203 (31). Las extensiones de registro especifican un procedimiento general para nodos HIP, que les permite suscribirse o registrarse en algún tipo de servicio o entidad intermedia. El servidor RVS es solamente uno de estos posibles servicios y en un futuro podrían incluirse otros, como por ejemplo, dispositivos firewall o NAT basados en HIP (este tema se trata ampliamente en el *draft* de Tschofenig (32)).

El procedimiento de registro consiste en un intercambio básico entre el cliente que desea registrarse y el servidor, intercambio al que se añaden una serie de parámetros. Primero, el servidor ofrece los servicios que puede prestar mediante un parámetro REG_INFO añadido al R1. A continuación, el cliente solicita, mediante un parámetro REG_REQUEST en el I2, que su HIT quede registrado en el servicio solicitado. Por último, el servidor podrá contestar

afirmativa o negativamente añadiendo un REG_RESPONSE o REG_FAILED respectivamente en el R2.

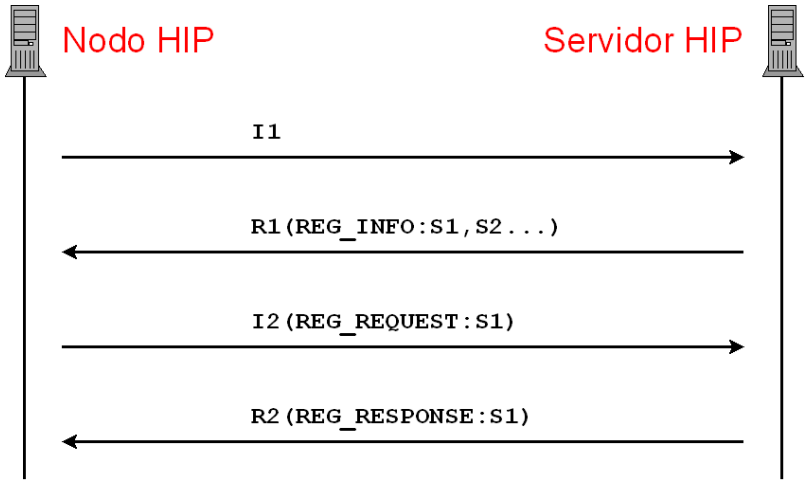

Ilustración 7: proceso de registro normal mediante un BEX.

Así mismo, un servidor puede actualizar la lista de servicios prestados a sus suscriptores utilizando paquetes UPDATE, de modo que estos pueden registrase usando una asociación HIP preexistente.

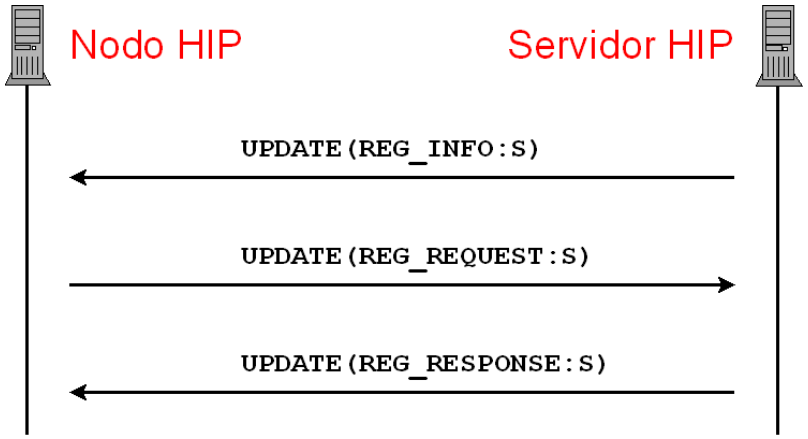

Ilustración 8: proceso de registro utilizando un intercambio de actualización.

A partir del momento del registro, un servidor estará informado de los posibles cambios de dirección IP de un nodo móvil mediante el, ya expuesto, procedimiento normal de actualización de asociaciones HIP. En el caso del servidor RVS, esto le permite mantener actualizada la relación HIT→IP de todos sus nodos registrados.

Cuando un determinado nodo desea recibir conexiones a través de un punto de encuentro, previamente debe publicar una relación de resolución entre él y su servidor RVS, es decir, que la dirección IP inicial asociada con el HIT del nodo sea la del servidor RVS. Esta relación sí puede mantenerse en un servicio de directorio como DNS, ya que no precisa cambios frecuentes. De hecho, sólo necesita modificarse en caso de que el nodo pase a utilizar otro RVS, ya sea por un cambio de proveedor de servicio o cualquier otro motivo. Dependiendo de si el servidor DNS incorpora las extensiones HIP (24) o se usa cualquier otro método de resolución, el procedimiento inicial puede diferir sustancialmente. Sin embargo, una vez que el iniciador ha obtenido la dirección IP del RVS, junto con el HIT y el HI del contestador, el intercambio básico se realizará siempre de la misma forma, tal y como se ve en el siguiente gráfico:

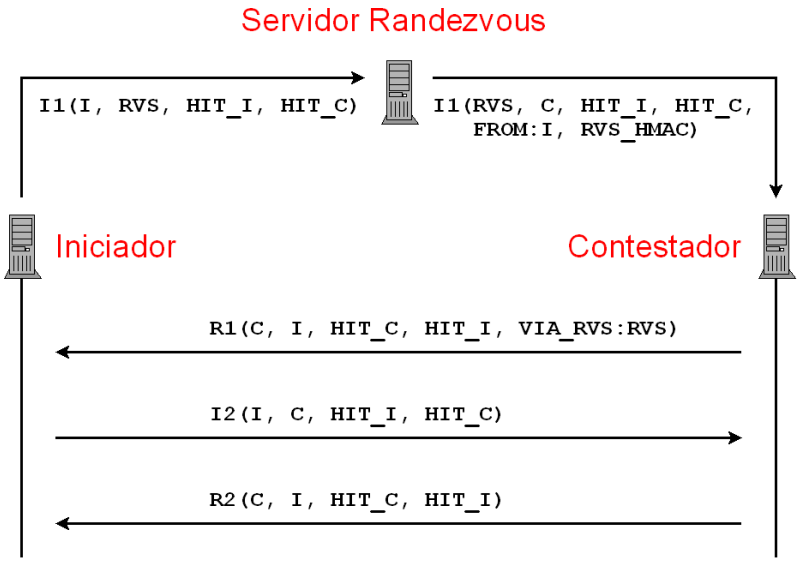

Ilustración 9: Intercambio Básico HIP mediante RVS.

Los cuatro primeros parámetros explicitados en cada paquete son: dirección IP origen, dirección IP destino, HIT origen y HIT destino. Como se puede ver, sólo el paquete I1 utiliza un método indirecto para alcanzar al contestador, pasando a continuación a una comunicación en modo directo entre los extremos. Se añaden, además, algunos parámetros nuevos.

El filtrado de salida (*egress filtering*), como mecanismo genérico de seguridad en redes públicas, impide que el RVS pueda reenviar los paquetes I1 recibidos sin sustituir la dirección IP del iniciador por la suya; esto hace necesario la inclusión de los parámetros FROM y RVS_HMAC. El primero contendrá la dirección IP del iniciador, y el segundo permite asegurar la integridad del parámetro anterior mediante la clave de integridad de la asociación establecida

entre RVS y contestador. Por lo demás, dado que el paquete I1 es un mero disparador del protocolo, no está protegido por parámetros HMAC o SIGNATURE, de modo que la manipulación del RVS no tiene mayores efectos de integridad extremo a extremo. El *checksum* IP, sin embargo, debe ser re-calculado por el RVS para que el contestador pueda verificarlo tal cual, sin necesidad de tener en cuenta el parámetro FROM.

Por último, las especificaciones generales del intercambio básico HIP exigen que, cuando se recibe un R1, se debe comprobar que éste responde a un I1 previamente enviado. Si la comprobación se limita al HIT no habrá problemas, pero fallará si se ésta se hace sobre la dirección IP. Por este motivo, cuando un contestador recibe un I1 a través de un servidor *randezvous*, deberá añadir a su R1 un parámetro VIA_RVS con la dirección IP del RVS implicado en el intercambio.

HIP A TRAVÉS DE NATS Y FIREWALLS

Las actuales especificaciones de HIP asumen que Internet proporciona un método directo para conectar con cualquier nodo. Sin embargo, en la Internet actual existe un creciente número de entidades intermedias (*middleboxes*) que realizan funciones que van más allá del simple reenvío de paquetes. Las entidades intermedias pueden dificultar el normal funcionamiento del protocolo HIP de dos maneras básicas: interfiriendo con el protocolo de control HIP o con el protocolo de transporte de datos ESP. En la RFC 3234 (33) se hace una extensa clasificación de estos dispositivos, pero los más comunes son, sin duda, los traductores de direcciones (NAT) y los cortafuegos (*firewall*).

El protocolo de control HIP, que incluye tanto los paquetes del intercambio básico como los de actualización de asociaciones y de notificación, utiliza distintos mecanismos para su transporte según se trate de IPv6 o IPv4. Mientras con IPv6 utiliza extensiones de cabecera (*extension headers*) especificas de HIP, con IPv4 usa el campo de datos IP (*IP payload*).

Los dispositivos NAT, aunque no representan un problema importante para IPv6, ya que su mayor espacio de direccionamiento los hace innecesarios, pueden plantear dificultades como elemento de transición entre IPv6 e IPv4. Por su parte, su uso en IPv4 está actualmente muy extendido. En el caso de un NAT básico, que realiza una simple traducción de direcciones, el trafico HIP puede atravesarlo sin ningún problema, ya que, tanto para IPv6 como IPv4, los *checksum* de nivel superior se calculan sobre una pseudo-cabecera IPv6 usando HIT como direcciones. Sin embargo, ese tipo de NAT básico no es el más común debido a que habitualmente un NAT también funciona como multiplexor IP[21], siendo habitual algún tipo de inspección y/o traducción de los puertos de nivel de transporte. Dado que el protocolo HIP no utiliza puerto alguno, dichos dispositivos no serán capaces de procesarlo correctamente, por ejemplo, para relacionar el tráfico de ida con el de vuelta. Además de esto, está el problema de que un nodo situado detrás de un NAT sólo puede actuar como iniciador por el conocido efecto de pérdida de direccionamiento directo.

Respecto a los cortafuegos, una política comúnmente recomendada en IPv4 es la de bloquear todo tráfico desconocido y permitir sólo algunos protocolos de transporte y, a menudo, sólo en determinados puertos. En el caso de IPv6, lo

[21] Esto sucede cuando una red IP con direccionamiento privado se conecta a Internet mediante una sola dirección IP pública.

previsible es seguir una política similar respecto a las extensiones de cabecera desconocidas. Otra política, ampliamente seguida, es la de bloquear la entrada en la red segura de todo tráfico que no sea identificable como respuesta a una conexión de salida previa. Esto ocasiona un problema similar al de los dispositivos NAT, limitando a los nodos de la red segura a actuar únicamente como iniciador.

El tráfico ESP, por su parte, cuando atraviesa un NAT experimenta problemas ya muy conocidos. Uno de ellos, la invalidación de los *checksum* de nivel superior cuando se utiliza el modo transporte, queda solventado por el uso del nuevo modo BEET, en el cual se utilizan las direcciones interiores para su cálculo (ver, ESP como transporte de datos en HIP). Sin embargo, ESP también carece de puertos, por lo que sufre el mismo problema ya descrito para el protocolo HIP. Aunque existe una propuesta, conocida como SPINAT (34), para utilizar el SPI como identificador de flujos en sustitución del puerto lo habitual es utilizar la solución propuesta en la RFC 3948 (35), conocida como NAT *Traversal* o NAT-T, y que consiste en encapsular el tráfico ESP en UDP usando un puerto determinado. La RFC 3947 (36) especifica como detectar la presencia de un NAT y como realizar la negociación de NAT-T mediante IKE.

En la RFC 5207 (37), se discuten todos estos aspectos de HIP relacionados con dispositivos NAT y cortafuegos, y se esbozan posibles soluciones. Algunas de ellas se desarrollan en otros documentos, como el *draft* de Tschofenig (32) que discute como extender la funcionalidad de dispositivos NAT y cortafuegos, de modo que puedan inspeccionar los mensajes HIP y derivar los flujos de datos subsiguientes.

Sin embargo, debido al enorme despliegue actual de estos dispositivos intermedios resulta impensable su actualización inmediata, por tanto, es necesario implementar soluciones que permitan el uso de HIP sin contar con su colaboración. En el *draft hip-nat-traversal* (38) se especifican una serie de extensiones al protocolo HIP que permite a dos nodos establecer una asociación y comunicarse, incluso en el caso de estar ambos tras un NAT o cortafuegos. De forma resumida, la propuesta incluye las siguientes funcionalidades:

- Encapsulado de los mensajes de control HIP en UDP, de modo similar al uso de NAT-T para ESP. La negociación para activar el encapsulado puede efectuarse durante el intercambio básico o mediante el mecanismo de actualización de asociaciones, ya que en cualquier movimiento un nodo puede caer indistintamente dentro o fuera de un

espacio de direccionamiento privado. De esta forma, para los nodos HIP que se encuentran tras un dispositivo NAT la información de localización será la IP pública del NAT más un puerto UDP de encapsulado.

- Negociación del encapsulado NAT-T para ESP. En el marco de IPsec este trabajo lo realiza el protocolo IKEv2, pero en el caso de HIP debe ser realizado por su protocolo de control.
- Introducción del servidor HIP *Relay*, un servicio similar al RVS que permite el contacto inicial con un nodo contestador que se encuentra tras un NAT o cortafuegos. A diferencia del servidor RVS que sólo reenvía el paquete inicial del intercambio básico, un servidor *Relay* reenvía todos los paquetes de control HIP. De esta forma, se establece una suerte de canal de control indirecto entre los nodos comunicantes, el cual les permite intercambiar la información de su localización para establecer un canal de datos en modo directo.
- Incorporación de un mecanismo para la detección de dispositivos intermedios, y la caracterización de las trasformaciones que realizan sobre el tráfico. Esto se realiza mediante pruebas de conectividad basadas en la metodología ICE[22] (39).

Por último, se considera también la posibilidad de que HIP utilice siempre el encapsulado UDP en IPv4 para todos los paquetes de control, o al menos para el primero de cada intercambio, contribuyendo a reducir la complejidad del protocolo.

[22] *Interactive Connectivity Establishment.*

SEGURIDAD DEL PROTOCOLO HIP

HIP aprovecha el nuevo paradigma de Identidad de Host para proporcionar una autenticación segura de nodos y un intercambio rápido de claves para ESP. Pero además, HIP también intenta limitar la exposición a varias clases de ataques de negación de servicio (DoS – *Denial of Service*) o de intermediación (MitM – *Man in the Middle*), como veremos a continuación.

Los ataques de DoS de agotamiento de recursos se aprovechan del coste de establecer un estado de protocolo en el contestador, cuando es mayor comparado con el del iniciador. HIP permite al contestador incrementar el coste de arranque del protocolo en el iniciador y hace un esfuerzo para reducirlo en el contestador. Esto se consigue haciendo que el contestador arranque el intercambio Diffie-Hellman autenticado en vez del iniciador, alargando a cuatro paquetes el intercambio básico HIP.

HIP soporta de modo opcional la negociación oportunista. Es decir, si un nodo recibe el inicio de una conexión de transporte sin negociación HIP, puede intentar forzar un intercambio HIP antes de aceptarla; esto implica un riesgo potencial de un ataque DoS para ambos nodos. Si el método para forzar el inicio de HIP es caro para ambos extremos, el atacante sólo necesitará falsificar un TCP SYN[23], metiendo a los dos sistemas en operaciones costosas. HIP evita este ataque haciendo que el contestador envíe un paquete simple HIP que puede estar pre-calculado. Como ese paquete es fijo y fácilmente reenviable, el iniciador sólo reaccionará a él si previamente ha iniciado una conexión con el contestador.

En HIP, las SA de IPsec se indexan por su SPI; la IP de origen se ignora siempre y la de destino también puede ignorarse. Por lo tanto, el protocolo ESP combinado con HIP resulta independiente de la dirección IP. Podría parecer que esto facilita las cosas a los atacantes pero ESP, mediante su protección contra replicas, está suficientemente protegido y la eliminación de la dirección IP no debería aumentar su exposición a los ataques DoS.

Es difícil defenderse de ataques MitM si no existe un tercero que proporcione autenticación, ya que un atacante con capacidad para interceptar los paquetes podría manipular todos los mecanismos del BEX. Sin embargo, HIP proporciona posibilidades para evitarlo. Si los HI se obtienen de una zona DNS segura o por

[23] Se denomina así al paquete que inicia el intercambio de establecimiento de una conexión TCP.

cualquier otro medio que proporcione autenticación, iniciador y contestador pueden obtenerlo y validar los paquetes HIP firmados. Sin embargo, el iniciador puede elegir usar un HI anónimo (no publicado), arriesgándose a un ataque MitM. Consecuentemente, un contestador podría optar por rechazar el intercambio HIP con iniciadores que utilicen HI anónimos.

Como nunca todos los nodos soportarán HIP, los mensajes ICMPv4 del tipo "*Destination Unreachable*, *Protocol Unreachable*" e ICMPv6 del tipo "*Parameter Problem*, *Unrecognized Next Header*", se hacen necesarios y ofrecen la posibilidad de un ataque DoS. En este caso no se usa un paquete HIP porque, una de dos, o tendría que tener un contenido único y por tanto difícil de generar propiciando otro ataque DoS más, o por el contrario sería tan fácil de falsificar como un mensaje ICMP. Contra un iniciador, el atacante puede responder al I1 con un mensaje ICMP falsificado aparentando que el contestador no soporta HIP, recibiendo el iniciador el R1 válido poco después. Por otro lado, contra el contestador se puede simular un rechazo administrativo al envío del R1 con un mensaje ICMP del tipo "*Destination Unreachable*, *Administratively Prohibited*". Así, para protegerse contra este tipo de ataques, los nodos HIP no deberían reaccionar a un mensaje ICMP hasta transcurrido un tiempo razonable, permitiéndole atrapar el paquete HIP del auténtico corresponsal. Si éste no llega, entonces el nodo tendrá que asumir que el paquete ICMP recibido era correcto. Los casos descritos son los únicos momentos en el BEX donde resultan apropiados estos mensajes ICMP, pudiendo ser ignorados en cualquier otro punto del intercambio.

LISTAS DE CONTROL DE ACCESO

El espacio de nombres HIP carece de la capacidad de agregación de las direcciones IP. Hoy en día, ésta propiedad no sólo se usa para consolidar la información de enrutamiento, sino también ciertos aspectos de la administración de sistemas. La típica lista de control de acceso (ACL[24]) utiliza máscaras para referirse a grupos de nodos de la misma subred que pueden acceder a determinado recurso. Dado que el valor de un HIT es un hash, al menos en parte, sólo podrá agregarse la porción asignada administrativamente con las consiguientes limitaciones operativas.

Sin embargo, un cortafuegos podría usar HIT para controlar la entrada/salida de una red con un nivel de garantía difícil de conseguir a día de hoy. Como ya se ha

[24] *Access Control List.*

dicho anteriormente, una vez establecida una sesión HIP el valor de SPI de los paquetes ESP puede usarse como índice referenciando a los HIT. En la práctica, un cortafuegos puede inspeccionar los paquetes HIP para aprender los vínculos entre HIT, SPI e IP. Incluso pueden controlar explícitamente el uso de IPsec, abriendo dinámicamente el paso de ESP sólo para direcciones IP y valores de SPI específicos. Las firmas de los paquetes HIP permiten a un cortafuegos capacitado asegurarse de que el intercambio HIP se lleva realmente a cabo entre dos nodos conocidos, lo que aumentaría sensiblemente la seguridad.

Ha habido experiencias negativas con listas de acceso distribuidas que contienen material relacionado con claves públicas, por ejemplo, con el protocolo SSH[25]. Si el dueño de una clave necesita revocarla por el motivo que sea, buscar todos los lugares donde una ACL incluye dicha clave puede ser una tarea imposible. Si el motivo de la revocación es el robo de la clave privada, esto podría suponer un serio problema. Un nodo HIP puede seguir la pista de todos los corresponsales que podrían estar usando su HIT en una ACL simplemente anotando los HIT de los contestadores con los que ha creado asociaciones. Con esa información el nodo podría enviarles un aviso de revocación de HIT, sin embargo, no se ha desarrollado aún un método seguro para la emisión de tales avisos.

Por otra parte, un dispositivo NAT con soporte HIP resulta técnicamente transparente para cualquier pareja de nodos que utilicen ese protocolo y, por tanto, un nodo puede tener dificultades para avisar a cualquier NAT que está usando un determinado HIT en sus ACL. Aunque la mayoría de sistemas conocerán la presencia de dispositivos NAT en sus redes, debería haber un proceso por el cual pudieran notificar a esos NAT de los cambios de HIT. Esto resulta fundamental para sistemas que actúan como contestador detrás de un NAT. Finalmente, si un nodo es avisado del cambio de HIT de un corresponsal, debería a su vez notificar el cambio a su NAT, de modo que éste permanezca actualizado.

USO DE IDENTIFICADORES NO CRIPTOGRÁFICOS

La definición de Identificador de Host establece que éste no tiene por qué ser necesariamente una clave pública, lo que implica que el HI podría ser cualquier valor, por ejemplo, un FQDN. Los HI no criptográficos aún ofrecerían los servicios de HIT o LSI y sería posible enviarlos en paquetes HIP, aunque no

[25] *Secure Shell Protocol.*

proporcionarían autenticación ni privacidad. Éste modo de operación no ha sido definido, ya que ofrecería muy pocas funcionalidades en comparación con las modificaciones necesarias del núcleo IP.

Por este motivo, HIP sólo debería implementarse usando Identificadores de Host de clave pública. Si fuera deseable usar HIP en un entorno de baja seguridad donde las operaciones de clave pública sean consideradas costosas, podría usarse HIP con claves muy cortas tanto para Diffie-Hellman como para el HI. Dicha práctica haría que los nodos participantes quedasen vulnerables frente a ataques de secuestro de conexión y MitM. Sin embargo, otros mecanismos como el de comprobación de direcciones, que se apoya en el enrutamiento y no en la fuerza del sistema criptográfico, seguirán siendo una protección frente a otro tipo de ataques.

BENEFICIOS DE HIP

Tradicionalmente, los protocolos de nivel red tenían cuatro invariantes clásicos:

- **Inmutabilidad**: las direcciones usadas por el emisor son las mismas que llegan al receptor.
- **Inmovilidad**: las direcciones no cambian durante el curso de una asociación.
- **Reversibilidad**: siempre puede formarse una cabecera de vuelta intercambiando las direcciones origen y destino.
- **Omnisciencia**: todo nodo sabe qué dirección usa su corresponsal para enviarle los paquetes.

En la actualidad nos hemos visto obligados a prescindir de la primera y la cuarta, debido fundamentalmente a las limitaciones de direccionamiento en IPv4 y el consiguiente uso de dispositivos NAT. De forma intencionada, también queremos deshacernos de la segunda invariante para conseguir movilidad y *multihoming*. La propuesta RSIP, ya mencionada anteriormente, es una alternativa a NAT que intenta restituir la cuarta invariante sin necesidad de la primera. Mientras que IPv6, por su parte, trata de restituir la primera.

HIP permite prescindir de forma real de todas las invariantes del nivel de red (que pasan al nivel de *host*) excepto la tercera. Esto es lo mismo que decir que, mientras las direcciones origen y destino en el protocolo de nivel de red sean reversibles, entonces todo irá bien; HIP se ocupa de la identificación de los extremos y la reversibilidad nos permite devolver los paquetes a nuestro corresponsal. No importa si las direcciones de nivel red cambian durante el trayecto o por el motivo que sea (son mutables o móviles), y tampoco importa cuál es la dirección de nivel red que está usando el corresponsal (no hay omnisciencia).

Por otra parte, hay pocos sistemas en Internet que tengan nombres DNS significativos. Es decir, si disponen de un FQDN éste pertenece habitualmente a un dispositivo NAT o un servidor de conexión telefónica (*dial-up server*), y no identifica al propio sistema sino a su conexión actual. Los FQDN suelen ser nombres de nivel de aplicación que frecuentemente se refieren a servicios más que a sistemas en particular. Este es el motivo de por qué muchos sistemas no están registrados en el DNS; no tienen servicios de interés para otros sistemas de Internet. Además, en último término estos nombres DNS sólo son

referencias a direcciones IP, lo que demuestra la fuerte interrelación actual entre los niveles de red y aplicación.

El espacio de nombres de Identidad de Host viene a llenar un importante hueco entre los espacios de nombres DNS e IP, independizándolos uno de otro. No solo permite identificar de forma univoca los extremos de una conexión, lo cual incorpora notables ventajas para la seguridad, sino que además puede hacerlo sin importar su localización en la red. De forma colateral, esto podría permitir al DNS pasar a ser un autentico espacio de nombres de nivel de aplicación, con poca o ninguna relación con el nivel de red.

RESPUESTAS DE HIP AL CUESTIONARIO DEL NSRG

El Grupo de Investigación sobre Espacios de Nombres (NSRG[26]) del IRTF, en su informe final de 2003 (3), propuso un cuestionario evaluador sobre la necesidad de nuevos espacios de nombres y sus beneficios potenciales. Debido a su valor como resumen estandarizado, se exponen a continuación las respuestas dadas por HIP en su RFC 4423 (4).

- **¿Cómo podría una Identidad de Host mejorar el funcionamiento general de Internet?**
  HIP desacopla los niveles de red y transporte, permitiendo que evolucionen por separado. Este desacoplamiento facilita la movilidad y el *multihoming*, también entre redes IPv4 e IPv6, simplifica la re-numeración del nivel red y hace más sencillo implementar migración de procesos y servidores en *cluster*.
- **¿Cómo es una Identidad de Host?**
  Un HI es una clave criptográfica pública. Sin embargo, en lugar de usar las claves directamente, la mayoría de los protocolos usan un hash de tamaño fijo generado a partir de la clave pública.
- **¿Cuál es su tiempo de vida?**
  HIP proporciona Identificadores de Host tanto estables como temporales. Los HI estables son habitualmente duraderos, con un tiempo de vida de años. El tiempo de vida de los HI temporales depende de cuánto tiempo los necesiten las conexiones de niveles superiores, y podrá oscilar entre segundos y años.
- **¿En qué parte de la pila se sitúa?**
  Los HI se situan entre los niveles de red y transporte.

[26] *Name Space Research Group.*

- **¿Cómo los usan los extremos?**
  Los Identificadores de Host pueden usarse directa o indirectamente (en la forma de HIT o LSI) por las aplicaciones cuando acceden a los servicios en red. Además, los Identificadores de Host, como claves públicas, se utilizan en el protocolo interno de establecimiento de clave, llamado intercambio básico HIP, para la autenticación mutua de los nodos.
- **¿Qué infraestructura es necesaria para su soporte?**
  En algunos entornos es posible usar HIP en modo oportunista, sin ninguna infraestructura. Sin embargo, para extraer el máximo beneficio de HIP, los HI deben almacenarse en el DNS o en alguna PKI. Para soportar movilidad es necesario, además, un nuevo mecanismo de punto de encuentro (*rendezvous*) que puede necesitar nueva infraestructura para su despliegue.
- **Si se añade un nivel nuevo ¿eliminaría la necesidad de la lista de direcciones en el protocolo SCTP?**
  Sí.
- **¿Qué beneficios ofrecería el nuevo espacio de nombres en materia de seguridad?**
  HIP reduce la dependencia de la dirección IP, haciendo más fáciles de solucionar los denominados problemas de propiedad sobre la dirección. En la práctica, HIP proporciona seguridad para la movilidad y el *multihoming*. Además, como los Identificadores de Hosts son claves públicas, las infraestructuras estándar de certificados de clave pública pueden ponerse en práctica por encima de HIP.
- **¿Cuáles serían los mecanismos de resolución, o qué características se requerían de un mecanismo de resolución?**
  En la mayoría de los casos, es suficiente con una aproximación donde los nombres DNS se resuelven simultáneamente a IP y HI. Sin embargo, si se necesita resolver un HI a una dirección IP o de vuelta a un nombre DNS, se hace necesaria una infraestructura de resolución plana. Tal infraestructura podría basarse en la idea de las Tablas Hash Distribuidas (DHT[27]), pero requería una significativa cantidad de desarrollo y despliegue.

[27] *Distributed Hash Table.*

PARTE III: TRABAJO REALIZADO

En este apartado se expone el trabajo práctico y experimental realizado con las implementaciones de HIP disponibles. El objetivo que se planteaba era comprobar la operatividad básica del protocolo, el estado de madurez de las implementaciones y, más concretamente, su capacidad para soportar la movilidad en IPv6. En este contexto se realizaron pruebas con diferentes aplicaciones nativas IPv6 que producen distintos patrones de tráfico, como son una sesión de terminal remoto, una transferencia de ficheros o una sesión de video *streaming*. Para ello se dispuso un escenario de pruebas que permitía simular movimientos en la red y al mismo tiempo controlar los parámetros necesarios de cada ensayo. La última prueba fue una comparación cuantitativa entre HIP y MIP acerca de la eficiencia en la gestión de la movilidad, es decir, el tiempo requerido para restablecer las conexiones tras el movimiento de un nodo.

IMPLEMENTACIONES DE HIP

Actualmente, existen tres diferentes implementaciones activas de código abierto del protocolo HIP. Éstas serán la base del trabajo en este proyecto y las veremos con más detalle a continuación. Las tres se actualizan regularmente con las últimas especificaciones publicadas y, en el marco del grupo de trabajo del IETF, los desarrolladores suelen realizar pruebas de interoperatividad para asegurar la validez del código. Existe, además, una cuarta implementación de código abierto escrita en lenguaje *Phyton* por Andrew McGregor, sin embargo, ésta se encuentra inactiva y ya obsoleta. También, algunos fabricantes de equipamiento y operadores de redes afirman tener implementaciones internas, pero dado su carácter propietario no hay información disponible sobre su estado actual o detalles de diseño.

HIP FOR LINUX (HIPL)

HIPL[28] es una implementación desarrollada por el *Helsinki Institute for Information Technology* (HIIT) y la *Helsinki University of Technology* (HUT) en

[28] http://infrahip.hiit.fi

Finlandia. Se encuentra disponible con licencia GPL[29], aunque puede conseguirse bajo demanda con una licencia menos restrictiva. HIPL, a su vez, está encuadrado en el proyecto *InfraHIP* que, integrado por instituciones académicas, empresas privadas (*Nokia, Ericsson*) y otros organismos públicos, tiene el objetivo de potenciar el desarrollo y despliegue de tecnología basada en HIP. Los integrantes del proyecto *InfraHIP* afirman que las especificaciones básicas del protocolo HIP ya están prácticamente listas y, por lo tanto, su trabajo está más orientado hacia otros aspectos necesarios para un despliegue generalizado, como pueden ser los servidores DNS y randezvous y el soporte para NATs y *firewalls*.

Como su propio nombre indica, HIPL está disponible principalmente para *Linux.* La implementación incluye un demonio "*hipd"* en espacio de usuario que trabaja junto con la implementación IPsec en espacio de *kernel*. Hasta hace poco tiempo era necesario parchear el *kernel* con objeto de incluir el modo BEET usado por HIP, sin embargo, a partir de la versión 2.6.27 de octubre de 2008, el código BEET ha sido incluido por defecto en todos los *kernel Linux*. Aunque también se liberan regularmente paquetes pre-compilados de HIPL con versiones numeradas (actualmente la 1.0.3), las actualizaciones del código son tan frecuentes que resultan poco útiles. Es mucho más recomendable usar la versión diaria del código fuente y compilarlo. Recientemente, también se inicio un proyecto para portar HIPL a *Symbian OS* y actualmente existen paquetes pre-compilados para los terminales *Nokia N800* y *N810*.

A pesar de su carácter experimental la documentación de HIPL es bastante completa, aunque a veces demasiado esquematica, incluyendo información aportada por colaboradores sobre diferentes aspectos de su experimentación. Por lo general está correctamente actualizada. Existe también una lista de correo electrónico bastante activa que ha resultado muy útil para resolver dificultades. A lo largo de las pruebas se detectaron un par de errores en el software de los que se informó a través de dicha lista.

OPENHIP

El proyecto OpenHIP[30] fue iniciado por la *Boing Phantom Works Company* en los EE.UU. y está disponible públicamente con licencia GPL. Además del desarrollo de HIP para los sistemas operativos más habituales

[29] *GNU General Public License.*
[30] http://www.openhip.org

(*Linux*, *BSD*, *MacOS* y *Windows*), este proyecto también desarrolla documentación y herramientas para la experimentación, como por ejemplo, *HIP Virtual World*, un entorno de pruebas virtual sobre VMware de muy fácil instalación que permite hacer ensayos con varios nodos HIP en un solo PC. Además, recientemente se ha liberado una primera implementación parcial de SHIM6 (40), un protocolo para la gestión de *multihoming* sobre la base de asociaciones HIP.

OpenHIP, es de entre todas las implementaciones actuales de HIP, la que ofrece un mayor abanico de posibilidades de instalación, disponiendo de varios sistemas operativos y de dos arquitecturas en espacio de usuario y de *kernel* respectivamente. Las dos arquitecturas incluyen un demonio, "*hip*", ubicado en el espacio de usuario que implementa la lógica de control del protocolo HIP. La diferencia, por lo tanto, entre ambas versiones realmente radica en la gestión de los datos de usuario.

La versión en espacio de usuario no requiere ningún cambio en el *kernel*, ya que todo el procesamiento relativo a IPsec se lleva a cabo en un proceso de usuario. Para ello se usa una ruta local que redirige los paquetes destinados a cualquier HIT (2001:10::/28) hacia un dispositivo virtual, el *Tap Driver*, (TUN/TAP en *Linux* y TAP-32 en *Windows*) que los envía al *socket* por el que escucha el demonio "hip". Para cursar el tráfico, este proceso selecciona la asociación HIP adecuada (estableciéndola previamente en caso de no existir) y realiza el encapsulado ESP antes de mandarlo por el interfaz de red.

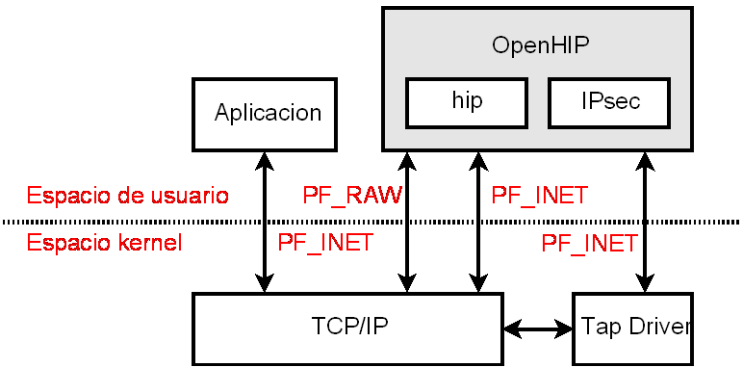

Ilustración 10: arquitectura de OpenHIP en espacio de usuario.

La desventaja de esta arquitectura es su bajo rendimiento, ya que toda la tarea de cifrado del tráfico ESP se debe implementar y ejecutar en espacio de usuario. Cada uno de los paquetes de usuario relativos a una asociación HIP tiene que ser copiado de la memoria *kernel* a la de usuario y repetir después la copia en dirección contraria. A pesar de este inconveniente, la instalación fácil y rápida

que proporciona resulta útil a efectos de experimentación. Además, esta arquitectura hace la portabilidad más sencilla y de hecho existen versiones pre-compiladas para *Linux, Windows y MacOS*.

Por otro lado, también se ha desarrollado una arquitectura con la parte de IPsec en espacio *kernel*, que sólo está disponible para *Linux*. Aquí, para desviar el tráfico destinado a cualquier HIT se utiliza una política IPsec en lugar de una ruta. Cuando se envía un paquete que encaja en dicha política el *kernel*, oportunamente modificado, informa al demonio "*hip*" que, a su vez, realiza el intercambio básico y establece una nueva SA añadiendo las correspondientes entradas en las tablas SAD y SPD[31] utilizando un socket PF_KEY. A continuación, cada nuevo paquete que el nivel de aplicación envía a ese HIT será encapsulado directamente por el *kernel*.

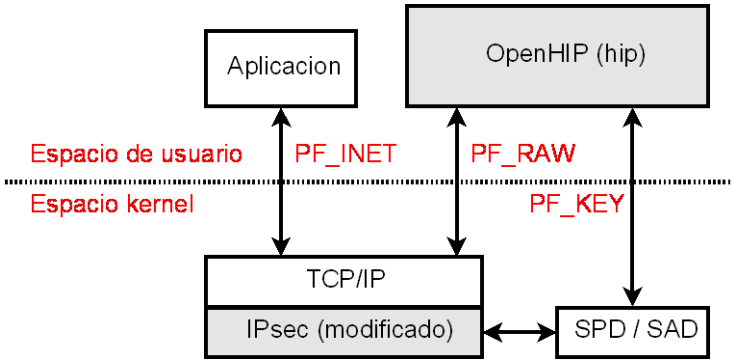

Ilustración 11: arquitectura de OpenHIP en espacio de *kernel*.

A pesar de que ésta arquitectura fue la primera elegida por este grupo de implementadores, después tomaron la decisión estratégica de priorizar la versión en espacio de usuario. El último parche liberado se publicó en el año 2006 para el *kernel* 2.6.13.5, una versión ya largamente desactualizada. Esto significa que, actualmente, esta versión está notablemente retrasada respecto a las últimas RFC y dispone de menos características. Por este motivo, en las pruebas posteriores usaremos sólo la versión en espacio de usuario.

El ritmo de liberación de nuevas versiones (actualmente 0.5.1) es lento pero regular, sin embargo, no se ofrece la posibilidad de acceso al código fuente diario. La documentación es bastante completa, aunque limitada a los aspectos básicos y desactualizada en algunas ocasiones. También existe una lista de correo para resolver problemas, sin embargo, parece muy poco activa.

[31] Las tablas SAD (*Security Association Database*) y SPD (*Security Policy Database*) son la forma estándar de IPsec para almacenar información. La SAD contiene las SA ya establecidas, mientras que la SPD describe el tráfico que debe ser protegido por IPsec.

HIP FOR INTER.NET (HIP4INTER)

Hip4inter[32], es un proyecto liderado por el *NomadicLab* de la compañía *Ericsson* en Finlandia y se encuentra protegido por una licencia especial conocida como "*Ericsson Finland Public Source*". Actualmente sólo existe operativa una versión para *FreeBSD*, aunque hay proyectos para su migración a *Linux* y *MacOS/X*.

Como en el caso de las anteriores implementaciones, esta consta de dos partes: un demonio "*hipd*" en espacio de usuario y un parche para el *kernel* BSD que incorpora el modo BEET al código de IPsec. Debido a esto, las versiones de Hip4inter se liberan regularmente para una versión especifica de *FreeBSD* (actualmente la 6.3), aunque existe la previsión de que el parche BEET pase a formar parte de la próxima versión oficial del *kernel* BSD.

Esta es la implementación que tiene una instalación más fácil y completa, incluyendo scripts de arranque automático del demonio y una solución muy simple para la resolución local de nombres. Pero por el contrario, es la implementación que proporciona menos funcionalidades, documentación e información en general.

[32] http://www.hip4inter.net

ESCENARIOS DE PRUEBAS

Para el desarrollo de las pruebas que nos habíamos propuesto primero fue necesario montar un escenario de pruebas incluyendo, no sólo los nodos HIP o MIP intervinientes, sino también una cierta infraestructura. La motivación era la de formar una pequeña topología de red que nos permitiera simular de forma sencilla los movimientos de un nodo móvil y, además, que nos diera libertad para la modificación de ciertos parámetros en los *routers*, en particular los del protocolo ND (*Network Discovery*) para la autoconfiguración de IPv6.

A lo largo del desarrollo de este trabajo el escenario de pruebas ha ido evolucionando según las necesidades. Aunque se han utilizado otros escenarios intermedios, los más representativos son los dos que se exponen a continuación. La diferencia básica entre ellos consiste en la utilización de máquinas virtuales o reales para los nodos principales (MN, CN y HA/RVS). Se puede encontrar información más específica sobre la configuración de estos escenarios en los apéndices correspondientes.

CON MÁQUINAS VIRTUALES

Para las primeras pruebas de instalación, intercambio básico y movilidad se consideró más práctico realizarlas sobre un escenario formado íntegramente por máquinas virtuales. Pese a las evidentes limitaciones de rendimiento, en este caso se optó por la virtualización dadas las siguientes ventajas:

- Posibilidad de clonar máquinas completas, ahorrando tiempo de instalación.
- Posibilidad de control a distancia de los nodos, facilitando el trabajo con un software poco estabilizado.
- Posibilidad de hacer *backups* y recuperaciones rápidas, muy útil en procesos de instalación poco claros.

Como soporte de virtualización se utilizó VMWARE SERVER 1.0.3 en una plataforma PC con doble procesador *Intel Xeon* y sistema operativo *Linux Debian Etch* con *kernel* 2.6.18.

En concreto, el escenario (ilustraciones 12 y 13) consta de una pareja de máquinas virtuales para cada sistema operativo necesario (*Linux* y *FreeBSD*), que hacen los papeles de MN y CN para cada una de las implementaciones de

HIP y MIP que se van a probar. Además, se instala otra máquina virtual encargada de hacer la función de RVS o HA, más otras dos que hacen el papel de *routers* de acceso (AR1 y AR2). Estas tres máquinas encaminan el tráfico de sus respectivas redes virtuales (vmnet 20, 21 y 22)[33] hacia otra red virtual (vmnet 10) que interconecta a todas ellas. El CN se encuentra conectado permanentemente a la vmnet 10, mientras el MN dispone de las otras tres redes virtuales para simular sus movimientos.

Para que todo este escenario esté conectado a su vez con Internet IPv6, se define un *bridge* que enlaza la red virtual vmnet 10 con la vlan 810[34], a la que está conectada la propia máquina anfitrión. Como infraestructura adicional se utiliza un *router Cisco VXR 7200* que encamina la vlan 810 hacia la red troncal de la UC3M, así como un conmutador *Cisco Catalyst 3500* para conectar todo el conjunto.

A continuación se muestran dos visiones de este escenario de pruebas, la primera muestra un punto de vista lógico o de nivel red, mientras que la segunda da una visión física o de nivel enlace.

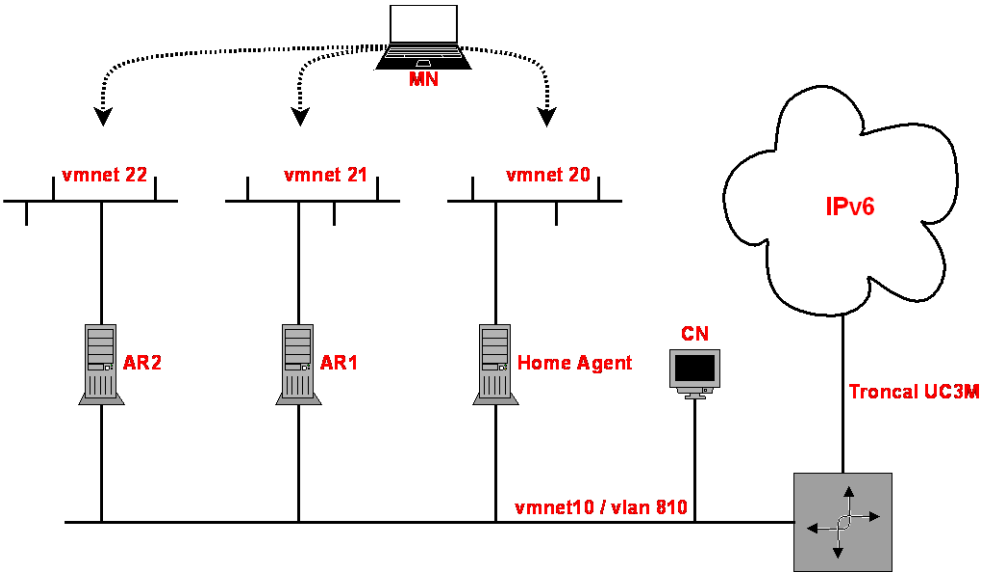

Ilustración 12: escenario de pruebas con máquinas virtuales (vista lógica).

[33] Las redes virtuales "vmnet" son simulaciones software de una LAN. Se trata de una herramienta incluida en el soporte de virtualización VMWARE, y su ámbito se limita a las máquinas virtuales que se ejecutan sobre dicho soporte. Es decir, una vmnet sólo puede conectar entre sí máquinas residentes en un mismo servidor VMWARE.

[34] Como "vlan", se entiende una red virtual definida mediante una infraestructura 802.1q. A efectos prácticos, para nuestras pruebas una vlan equivale a una LAN física.

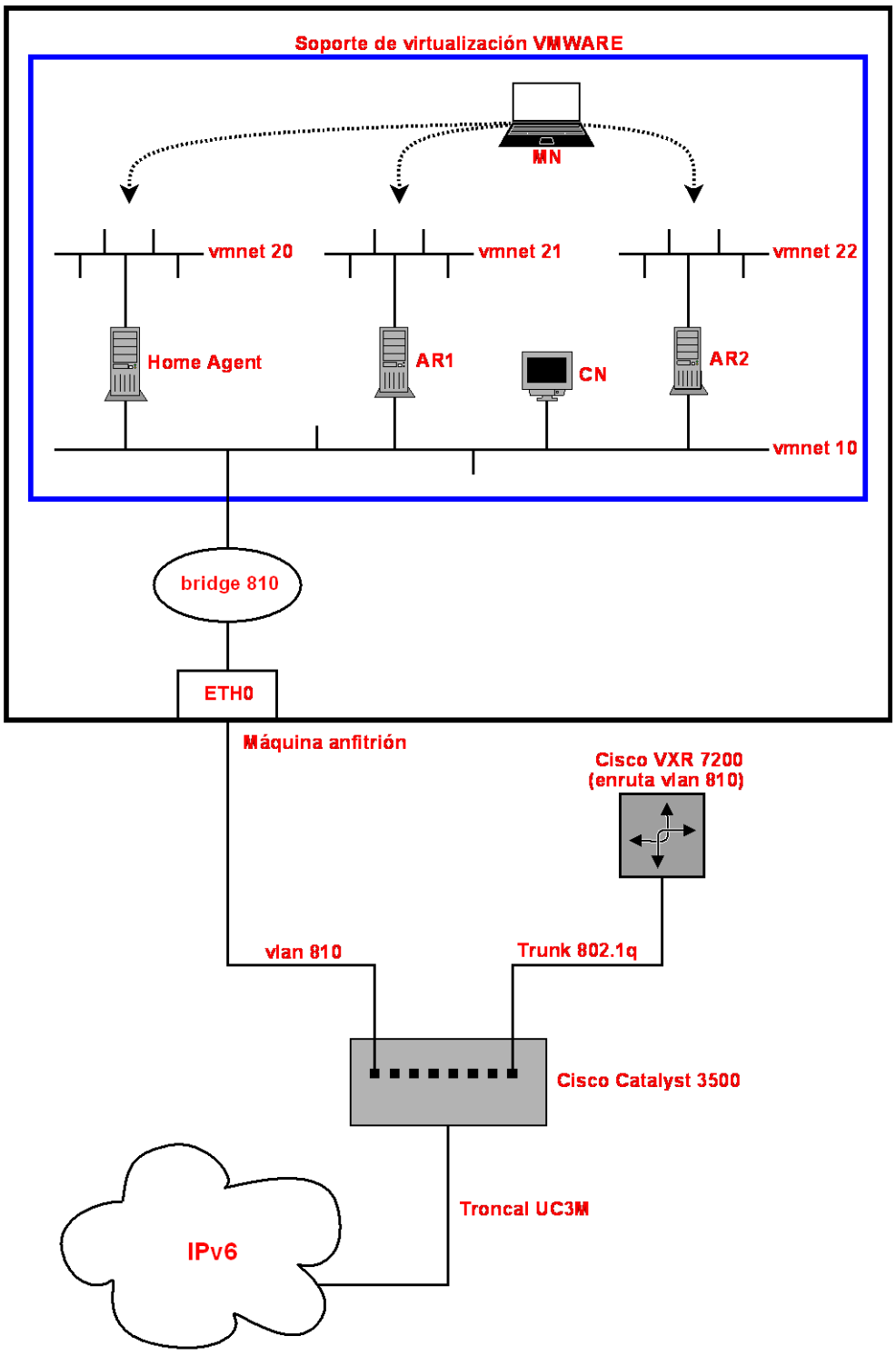

Ilustración 13: escenario de pruebas con máquinas virtuales (vista física).

Los *routers* de acceso, incluido el HA, utilizan como sistema operativo *Linux Debian Etch* con *kernel* 2.6.18. Como software de enrutamiento se optó por

Quagga[35], ya que su sintaxis de configuración es muy similar a la de *Cisco IOS* y es de fácil instalación. Además, también es necesario instalar *radvd*[36], que realizará el envío de los anuncios RA (*Router Advetisement*) para la autoconfiguración IPv6.

CON MAQUINAS REALES

Pese a que el escenario anterior es útil para familiarizarse con el software a prueba, el pobre rendimiento de las máquinas virtuales dificulta en gran medida la realización de las pruebas posteriores. Además, para experimentar el funcionamiento de HIP con algunas aplicaciones como *vlc*[37], es necesario contar con entorno gráfico en el nodo móvil. Los intentos realizados demostraron que dichas aplicaciones no conseguían funcionar correctamente sobre una máquina virtual, motivo por el cual se decidió sacar este nodo a una máquina real.

Además, otra de las pruebas, la medición del tiempo de re-direccionamiento, resulta poco fiable en un escenario de máquinas virtuales. Los tiempos que se obtuvieron a lo largo de algunos intentos eran altos y con amplias variaciones. Aunque no se pretendía con esta prueba hacer mediciones precisas, se consideró que era necesario que todos los nodos implicados en la re-dirección (MN, CN y HA/RVS) estuvieran en maquinas reales para poder hacer mediciones que fueran, al menos, más realistas.

Para adaptar nuestro escenario de pruebas, en el conmutador se añaden tres nuevas vlanes, la primera (820) para la *Home Network* que se conecta a través del HA y la vlan 810, y otras dos (811 y 812) que servirán para simular los movimientos del MN. Ahora el *router* deberá encaminar las vlanes 811 y 812, además de la 810 que ya encaminaba en el escenario anterior.

Con este nuevo escenario se pueden hacer todas las pruebas previstas, excepto una. El tiempo medio entre RAs es un parámetro fundamental en las mediciones del tiempo de re-direccionamiento tanto HIP como MIP. Cisco IOS,

[35] "*Quagga Routing Suite*" (http://www.quagga.net/) es un software de código abierto que permite a un PC realizar las funciones de un *router*. Implementa los protocolos de intercambio de rutas (*routing*) más comunes para una fácil integración en una red de *routers*.
[36] "*The Router Advertisement Daemon, radvd*"(http://www.litech.org/radvd/) es un software de código abierto que implementa el protocolo ND (*Network Discovery*) de IPv6.
[37] "VLC Media Player", también conocido como "VideoLAN" (http://www.videolan.org/vlc/) es un reproductor multimedia de código abierto. Implementa, además, las funciones de servidor y cliente de *streaming*, lo que permite el envió de audio y video a través de una red IP.

el sistema operativo de los *router* Cisco, no permite establecer valores para este parámetro por debajo de 500 ms, limitación que impide hacer una de las mediciones previstas con un tiempo medio entre RAs de 60 ms. Lo que nos lleva a la necesidad de reintroducir los *routers* de acceso con "*radvd*", cuya implementación del protocolo ND sí admite todos los valores que necesitamos. Sin embargo, estos nodos pueden seguir siendo virtuales a pesar del bajo rendimiento, ya que el único efecto será un mayor RTT entre MN y CN (que a su vez simula una mayor distancia topológica entre los nodos). El siguiente gráfico ilustra el escenario de pruebas definitivo.

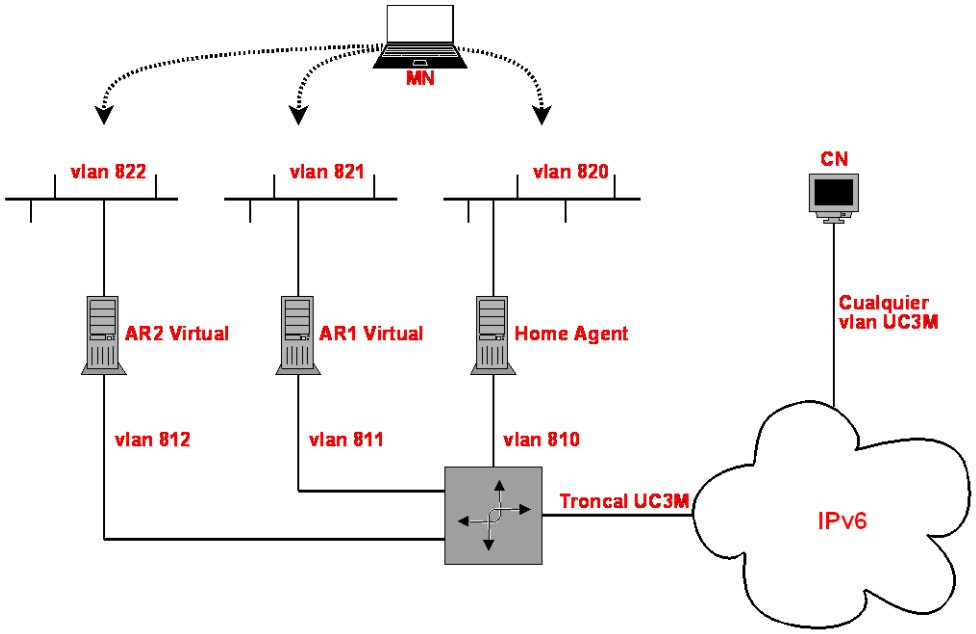

Ilustración 14: escenario de pruebas con máquinas reales (vista lógica).

Para integrar los nodos virtuales AR1 y AR2, se crean dos nuevas vlanes, 821 y 822, que encaminan su tráfico a través los *routers* de acceso hacia las vlanes 811 y 812. Para implementar esto en la máquina anfitrión, su interfaz de red se configura en modo *trunk* 802.1q, de modo que pueden definirse sub-interfaces para cada una de las vlanes que deben conectarse a los *routers* de acceso. Mediante bridges locales estas vlanes se conectan con las redes virtuales de VMWARE de la misma forma que ya vimos en el escenario anterior. El siguiente gráfico muestra una visión física, o de nivel de enlace, del escenario de pruebas descrito

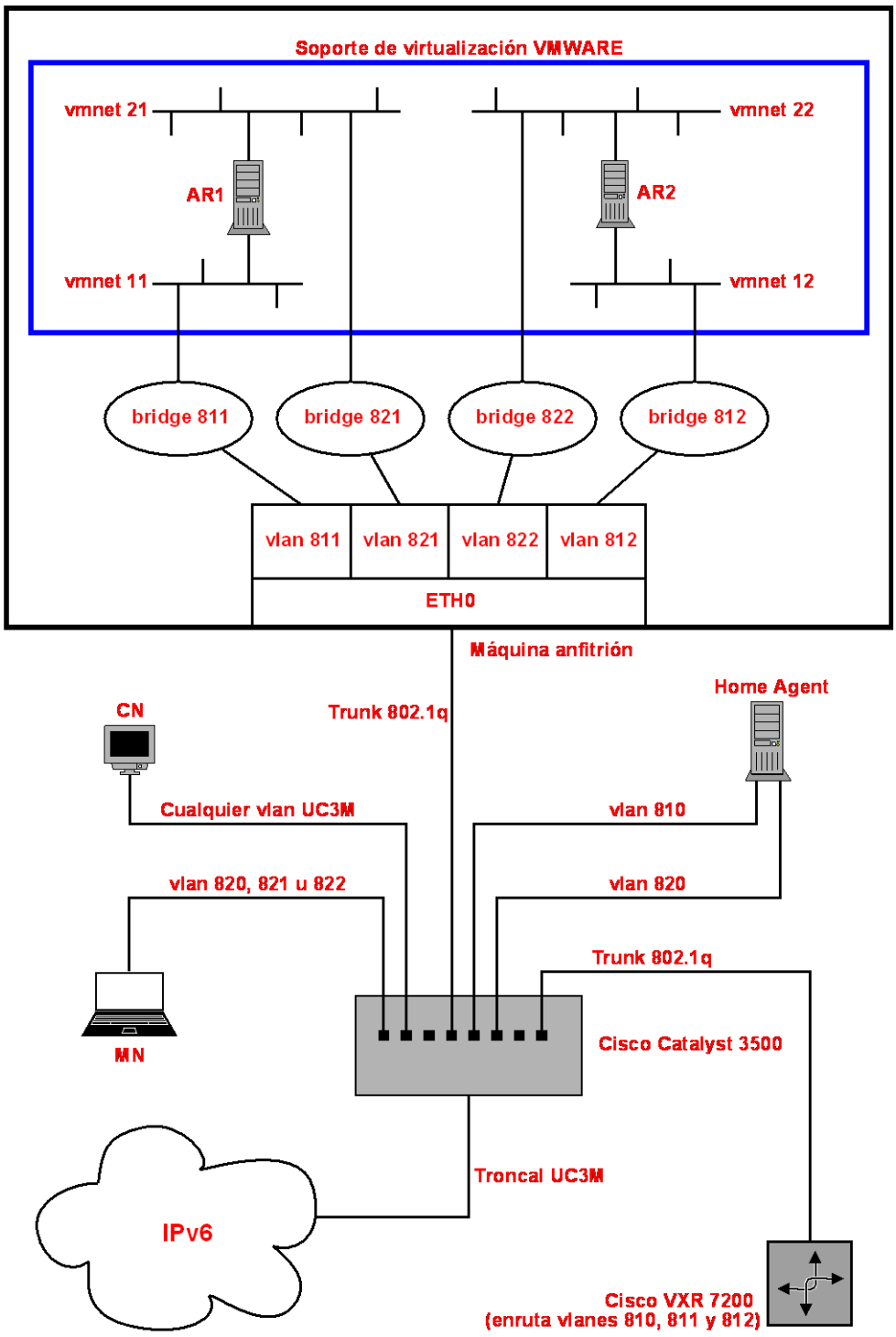

Ilustración 15: escenario de pruebas con máquinas reales (vista física).

En este escenario el CN puede estar conectado a cualquier vlan IPv6 de la UC3M, mientras que el MN simula su movimiento cambiando la vlan asociada a su puerto en el conmutador de acceso, quedando instantáneamente conectado a la subred correspondiente. Sin embargo, hay que tener en cuenta que, para evitar la formación bucles de nivel de enlace, cualquier puerto de un

conmutador *Ethernet* se comporta como un *bridge* y por tanto implementa el protocolo *Spaning Tree* (STP, IEEE 802.1D). Esto significa que cuando un puerto se suma a una vlan, dicho puerto debe pasar por los estados *listening* y *learning* antes de llegar al estado *forwarding*, momento en el cual el nodo conectado empezará a recibir tráfico y podrá cursar sus paquetes hacia la red. El tiempo necesario para llegar al estado *forwarding* es aproximadamente 30 segundos, o sea, un tiempo llamado *forward delay* (15 segundos) por cada cambio de estado. Para reducir el retraso en la re-conexión del nodo tras un cambio de vlan se puede bajar el tiempo *forward delay* lo más posible (en el caso de un conmutador *Cisco*, el mínimo es 4 segundos, lo que supone un retraso total de 8 segundos). La mejor solución es utilizar una modificación del protocolo STP disponible en algunos conmutadores *Ethernet* conocida como *port-fast*[38]. Esta permite a los puertos de acceso pasar directamente al estado *forwarding* tras unirse a una vlan y proporciona al nodo conectado la posibilidad de emisión y recepción sin ningún retardo añadido. Esta solución, al estar disponible en el conmutador de nuestro escenario, fue la que nos permitió hacer una simulación de movimiento prácticamente instantáneo.

[38]Recogida por el IEEE en la norma 802.1D-1998 (suplemento 802.1w).

PRUEBAS REALIZADAS

INTERCAMBIO BÁSICO

Como paso previo al comienzo del intercambio básico es necesario hacer la resolución del HIT a una IP inicial. Si se desea partir de un FQDN será necesaria, además, otra resolución previa a la anterior mediante la cual se obtendría el HIT correspondiente. En todas las implementaciones disponibles se experimenta con distintas posibilidades para hacer estas resoluciones, como DNS, DHT, o i3. El mecanismo de Servidor *Randezvous* (RVS) es una solución propuesta por el propio protocolo HIP para obtener la IP actual de un nodo móvil a partir de su HIT, pero nada dice de la resolución a partir de un FQDN. El RVS es el único método que implica directamente al protocolo HIP, por lo que será el único método de resolución que examinaremos. Para el resto de pruebas usaremos la resolución local, es decir, mediante ficheros editados manualmente.

En todas las implementaciones, para lanzar el Intercambio Básico es necesario que se produzca tráfico IP que tenga HIT como direcciones de origen y destino. Cualquier aplicación puede hacer eso, tanto si está basada en IPv4 como en IPv6, mediante el uso de HIT o LSI respectivamente como si fueran direcciones IP. En las pruebas sólo utilizaremos aplicaciones IPv6 nativas, usando HIT en lugar de direcciones IPv6. En concreto, para el disparo del intercambio básico usaremos el comando *ping6*.

HIPL

En el nodo corresponsal, que hace el papel de contestador (*responder*), arrancamos el demonio “hipd” (“hipd –b”, si queremos ejecutarlo en segundo plano). En primer lugar, este demonio examina el directorio “/etc/hip” en busca de los HI locales y si estos no existieran generaría unos nuevos. A continuación, introduce una serie de cambios en la tabla de interfaces, la tabla de rutas local y la SPD de IPsec, las cuales detallaremos a continuación.

Por un lado, crea un nuevo interfaz, *dummy0*, al que asigna como dirección IPv6 el HIT correspondiente los HI del nodo. En versiones anteriores HIPL creaba cuatro HI: una pareja de HI público y anónimo usando criptografía RSA y otra

pareja similar usando criptografía DSA. Teóricamente, un nodo puede disponer de diferentes identidades que podrá usar indistintamente según su propia política. Esto exige una gestión local que asocie las distintas identidades disponibles con los servicios y/o corresponsales con los que se abran nuevas conexiones. Dicha gestión se encuentra aún poco desarrollada por lo que, debido a los problemas que creaba en la realización de pruebas, el número de HIT utilizados ha sido reducido por los desarrolladores a uno sólo[39]. En el futuro, cuando la parte del código afectada esté más probada, se volverá a incluir la posibilidad de disponer de varios identificadores.

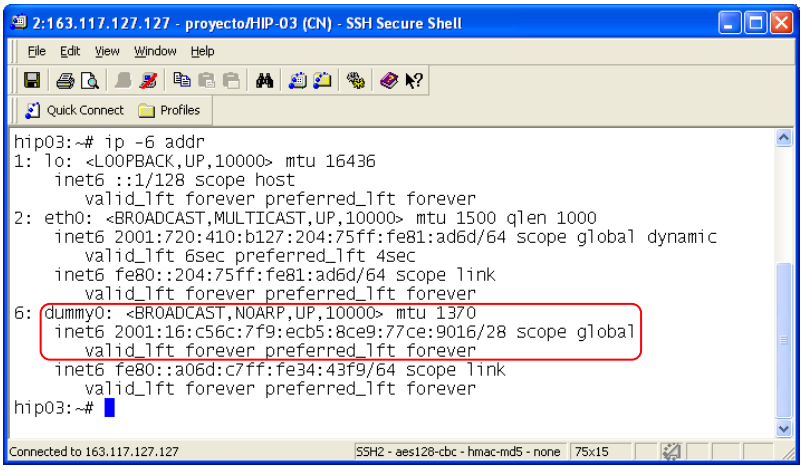

Ilustración 16: tabla de direcciones con HIPL.

Por otro lado, añade las rutas necesarias para que el tráfico correspondiente al prefijo 2001:10::/28 (el espacio de direccionamiento HIP) se curse a través del interfaz *dummy0*. Este interfaz virtual sirve para aislar las conexiones HIP de cualquier cambio en los interfaces físicos.

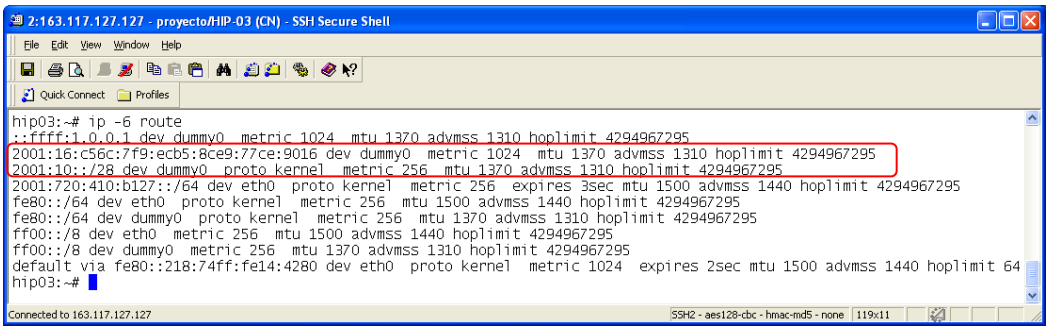

Ilustración 17: tabla de rutas con HIPL.

[39] Al funcionar como iniciador HIPL utilizaba siempre un HIT por defecto que podía obtenerse mediante el comando "*hipconf get hi default*". Para hacer los mapeos HIT/IP había que tener la precaución de usar siempre el HIT por defecto, pues de lo contrario surgían errores no siempre fáciles de depurar. Sin embargo, este sistema de HIT por defecto tampoco parecía funcionar siempre de una forma predecible.

Y por último, se insertan en la SPD las reglas que permiten identificar el tráfico HIP. Cuando no existe una SA para enviar el tráfico seleccionado por estas reglas, el demonio “hipd” recibe una notificación mediante un socket PF_KEY, disparando el intercambio básico.

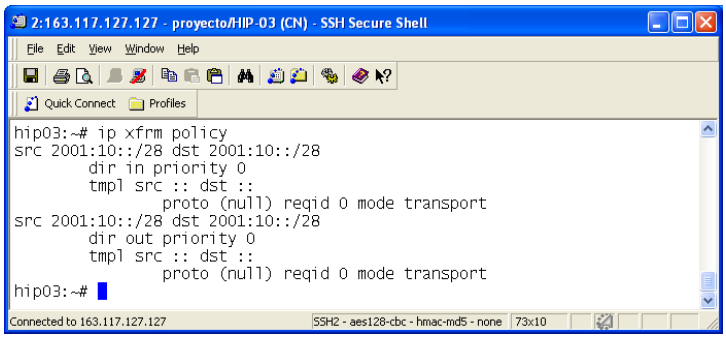

Ilustración 18: tabla SPD antes de realizar el Intercambio Básico con HIPL.

Situados ahora en el nodo móvil que hace el papel de iniciador (*initiator*), arrancamos también el demonio “hipd” y, mediante un *ping,* generamos tráfico hacia el HIT del nodo corresponsal. Si el intercambio básico falla y por tanto el *ping*, por lo general hay que reiniciar los demonios en ambos extremos ya que estos no son capaces de purgar las conexiones incompletas. Si, por el contrario, el intercambio básico tiene éxito, veremos que el *ping* funciona de la manera habitual, excepto por las direcciones que maneja.

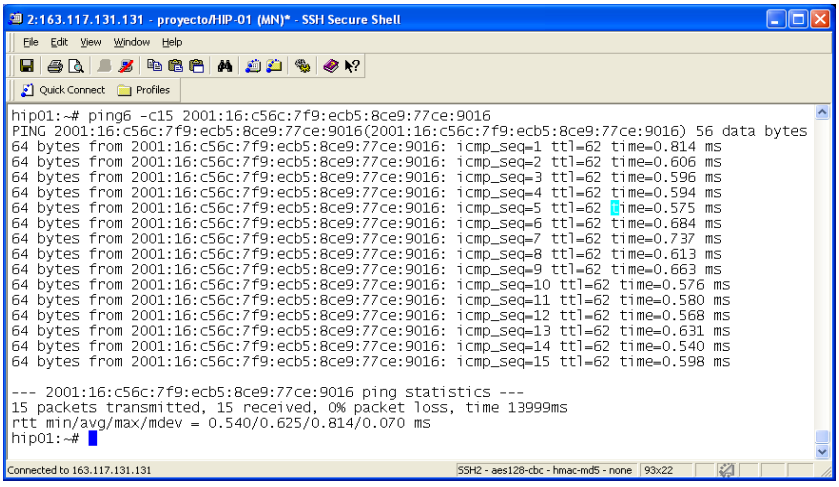

Ilustración 19: comando *ping6* usando un HIT en lugar de IP con HIPL.

Para la resolución del HIT del contestador en una dirección IP inicial, se utilizan los ficheros “/etc/hosts” (que relaciona nombre e IP) y “/etc/hip/hosts” (que relaciona nombre y HIT). Utilizando el mismo nombre en ambos ficheros se puede relacionar un HIT con su IP.

Puede observarse como el RTT del primer paquete es algo mayor que el de los siguientes. En contra de lo que inicialmente podría suponerse, esto no es debido al Intercambio Básico sino a algo relacionado con la activación de los túneles ESP. Haciendo una captura de este tráfico[40] desde el nodo móvil podemos ver lo que sucede. En la siguiente ilustración se ven destacados en verde los cuatro paquetes del intercambio básico, y en naranja los paquetes ESP con numero de secuencia igual a 1. Así, podemos ver como el Intercambio básico dura unos 200 ms, un tiempo muy superior al RTT del primer paquete.

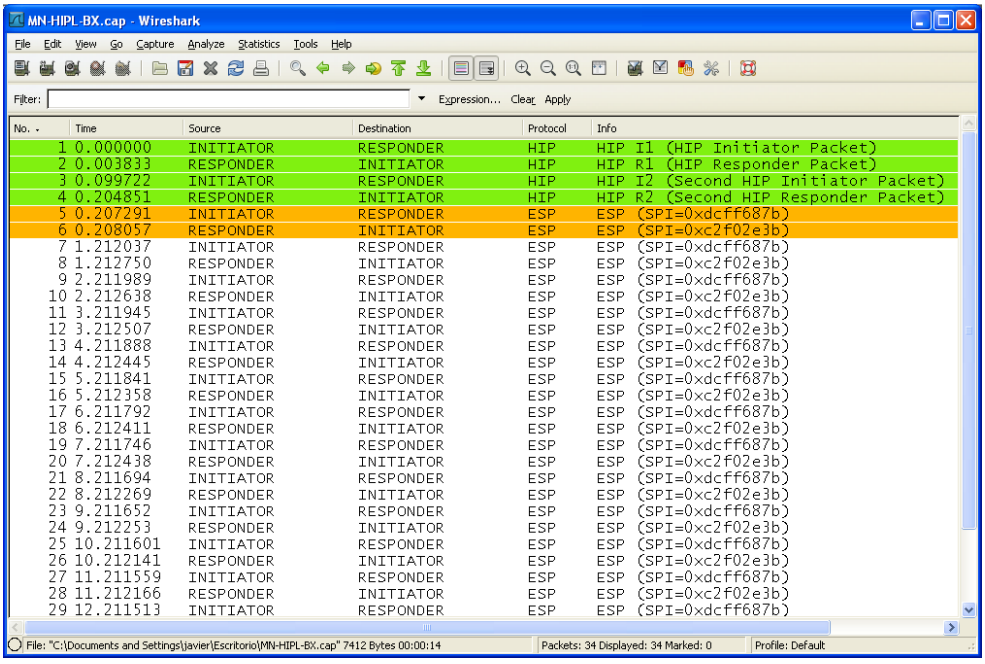

Ilustración 20: captura de tráfico de un BEX.

A través de las siguientes ilustraciones podemos ver el BEX más en detalle. En el paquete I1 observamos la cabecera IPv6 con las direcciones reales de cada nodo y la cabecera HIP con los HIT correspondientes. Además, en la cabecera IPv6 vemos que se usa como *Next Header* el valor 0x8b (139 en decimal), asignado recientemente por el IANA como número de protocolo específico para HIP[41].

[40] Para realizar capturas de tráfico se ha utilizado la herramienta *tshark* para Linux. Los ficheros capturados se reproducen aquí mediante una versión modificada de *Wireshark* para *Windows* capaz de decodificar el protocolo HIP. Dicha versión puede obtenerse a través de la página web de OpenHIP (http://www.openhip.org).

[41] Hasta hace poco se utilizaba el valor 0xfd (253 en decimal), un valor reservado para experimentación con nuevos protocolos. El analizador de protocolos utilizado no está aún actualizado por lo qué lo decodifica como *unknown*.

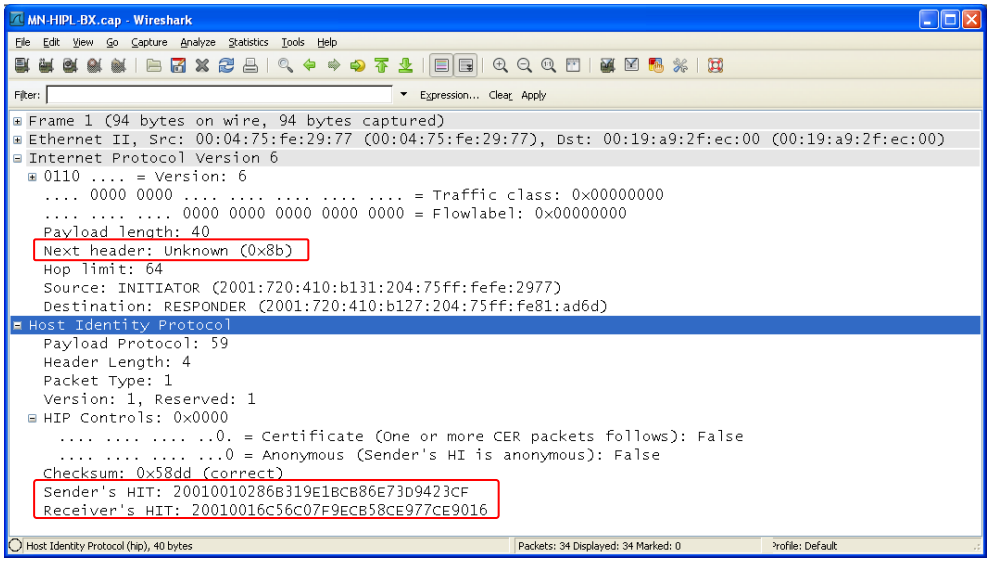

Ilustración 21: detalle del paquete I1 con HIPL.

En el paquete R1 se pueden ver, entre otros, los parámetros que contienen el puzle y el identificador HI del nodo móvil (parámetros PUZZLE y HOST_ID).

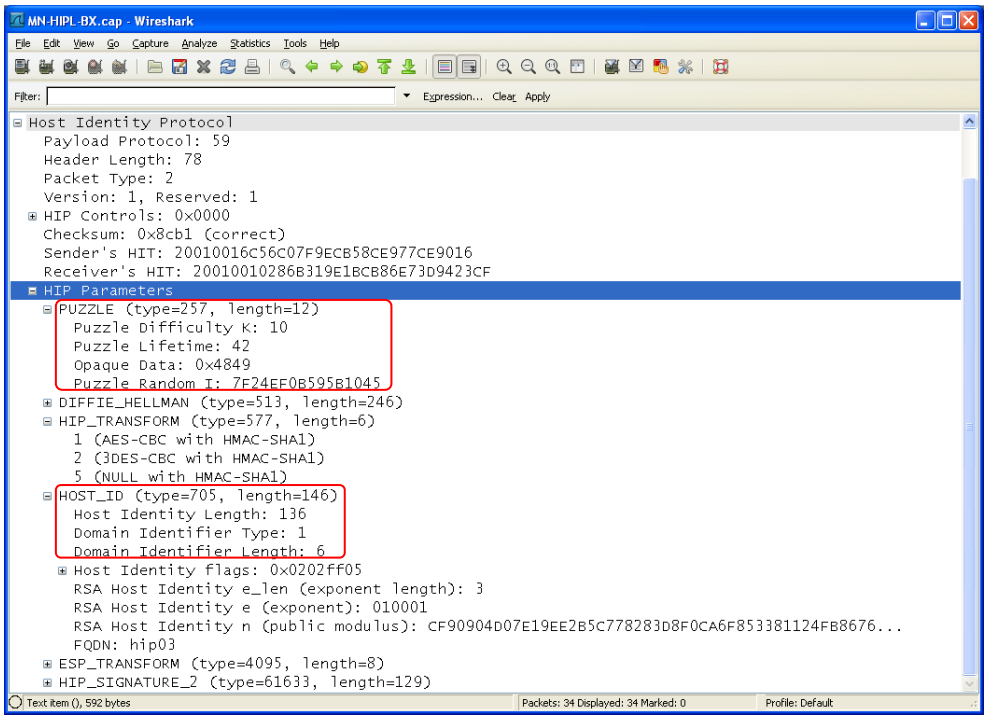

Ilustración 22: detalle del paquete R1 con HIPL.

En el paquete I2, tenemos la solución del puzle y el SPI correspondiente a la SA establecida en el iniciador con su contestador (parámetros SOLUTION y ESP_INFO). Además, podemos ver como el identificador HI del iniciador viene

cifrado en el parámetro ENCRYPTED, a diferencia de lo que sucedía en el paquete anterior.

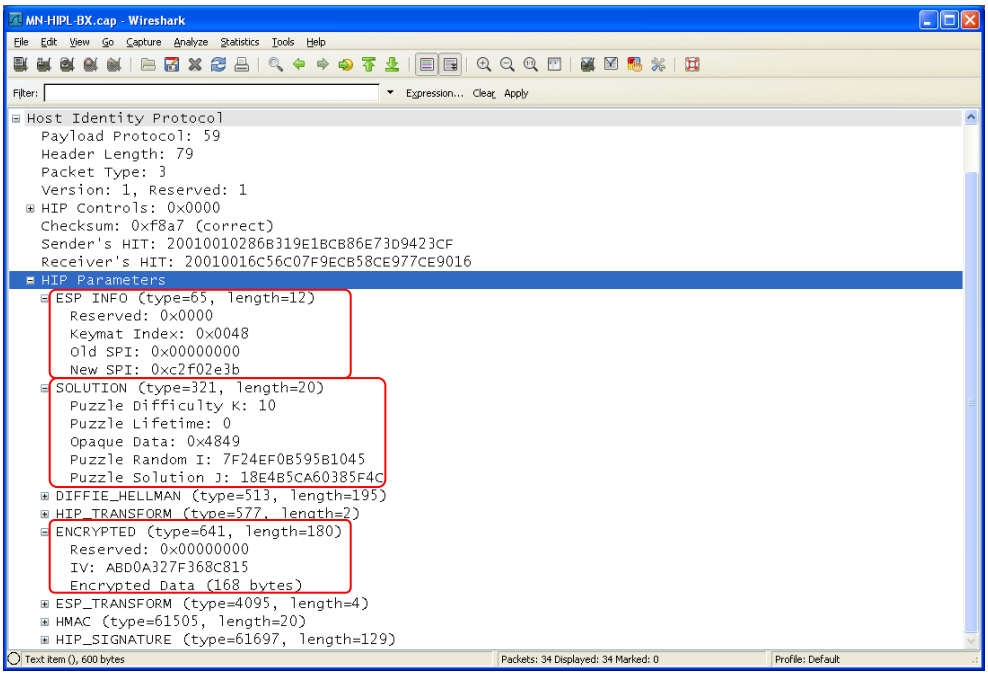

Ilustración 23: detalle del paquete I2 con HIPL.

Por último, en el paquete R2 vemos el SPI correspondiente a la SA establecida en el contestador con su iniciador.

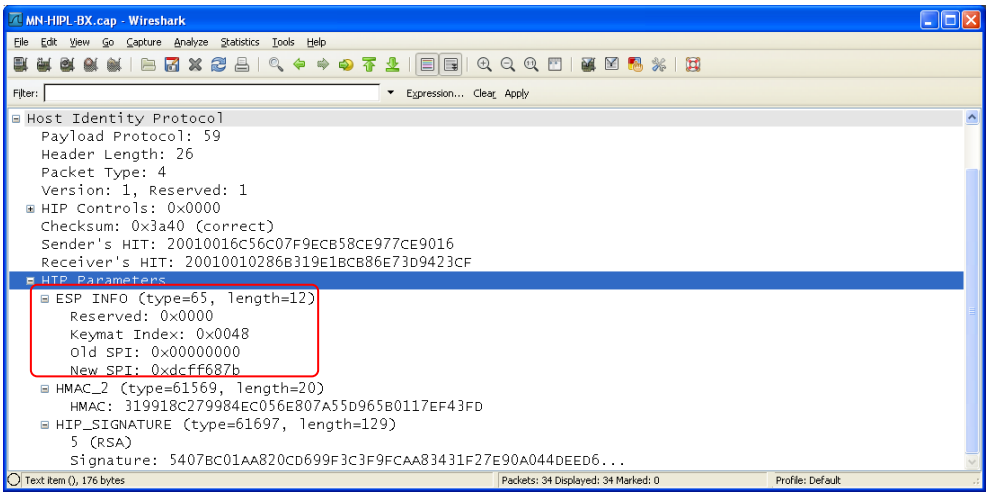

Ilustración 24: detalle del paquete R2 con HIPL.

Después de que el intercambio básico se haya realizado, podemos ver como se han añadido las reglas oportunas para dirigir el tráfico a las SA creadas al efecto. Puede observarse que dichas reglas y SA están definidas como "*mode*

4”, en referencia al nuevo modo BEET de IPsec, que aún no está correctamente decodificado por los comandos *ip*.

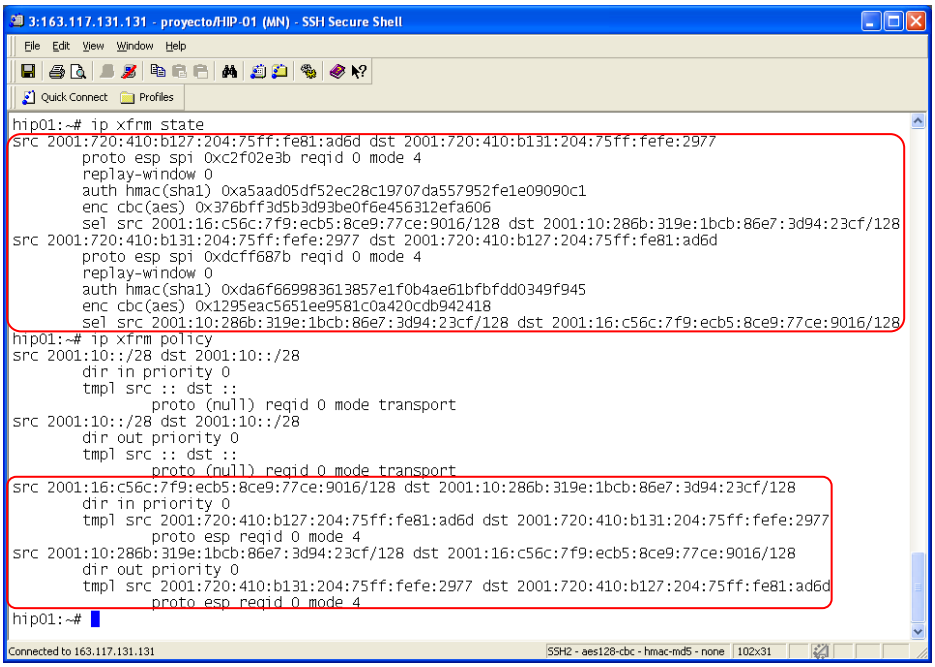

Ilustración 25: tablas SAD y SPD tras el Intercambio Básico con HIPL.

OPENHIP

En el nodo corresponsal, que hace el papel de contestador, arrancamos el demonio “hip –v” (“hip &> hip.log &”, si queremos ejecutarlo en segundo plano). En primer lugar, este demonio examina el directorio “/etc/hip” en busca del HI local y, a continuación, introduce una serie de cambios en la tabla de interfaces y la tabla de rutas local, que detallamos a continuación.

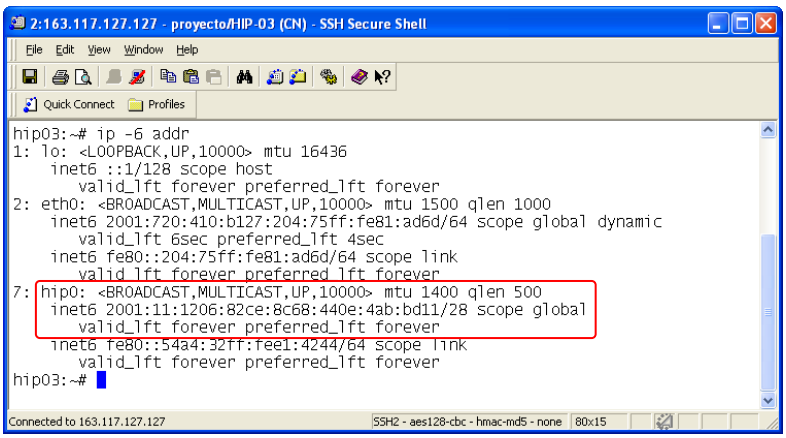

Ilustración 26: tabla de direcciones con OpenHIP.

Por un lado, añade un nuevo interfaz virtual, “hip0”, al que asigna como dirección IPv6 el HIT correspondiente al HI del host. Y por otro lado, añade las rutas necesarias para que el tráfico correspondiente al prefijo 2001:10::/28 (el espacio de direccionamiento HIP) se curse a través del interfaz hip0. Este interfaz sirve para aislar las conexiones HIP de cualquier cambio en los interfaces físicos.

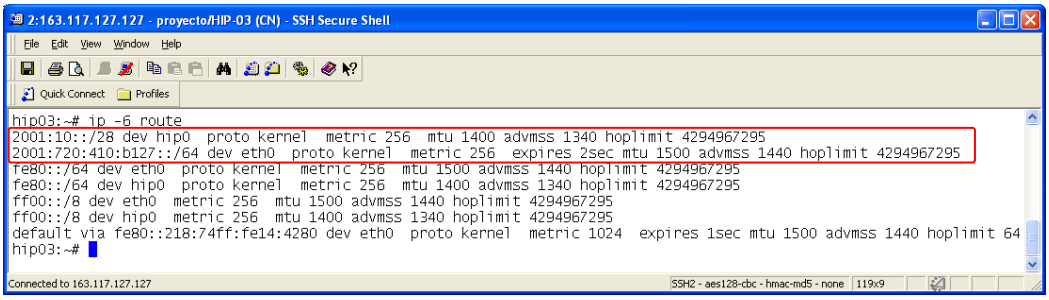

Ilustración 27: tabla de rutas con OpenHIP.

Situados ahora en el nodo móvil, que hace el papel de iniciador, arrancamos el demonio “hip” y, mediante un *ping*, generamos tráfico hacia el HIT del nodo móvil. Para conseguir la resolución del HIT del contestador en una IP inicial, es necesario editar el fichero “/etc/hip/known_hosts.xml” añadiendo una nueva instancia para cada nodo. Si el intercambio básico tiene éxito, veremos que el *ping* funciona de la manera habitual, excepto por las direcciones que maneja. Como en el caso de HIPL, el RTT del primer paquete es mayor debido a que incluye el tiempo de activación de los túneles ESP. Pero además, vemos que el RTT de todos los paquetes es mayor comparado con el de HIPL, debido al bajo rendimiento de la arquitectura en espacio de usuario de OpenHIP.

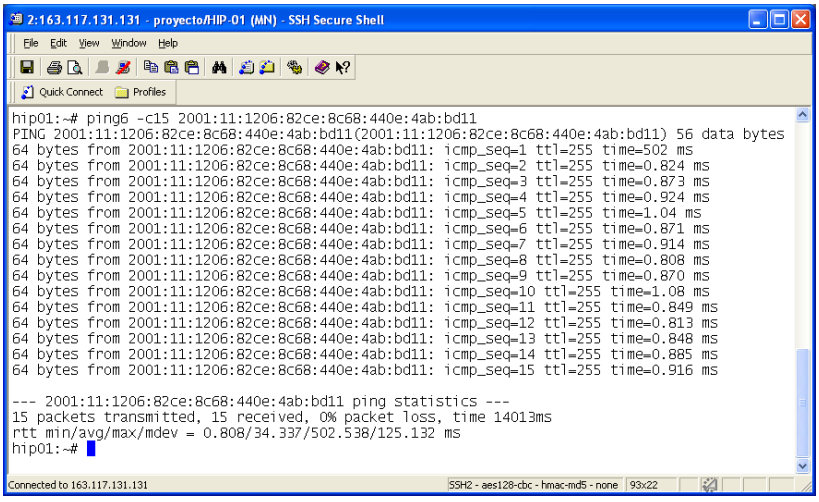

Ilustración 28: comando *ping6* usando un HIT en lugar de IP con OpenHIP.

Cuando se produce un fallo durante el BEX es necesario reinicializar los demonios, además como la lectura del fichero de "known_hosts.xml" se hace sólo al arrancar el demonio, cualquier cambio en este fichero exige su reinicio, junto con las demás conexiones activas. Si se mata el demonio es necesario borrar manualmente el fichero de proceso "/var/run/hip.pid".

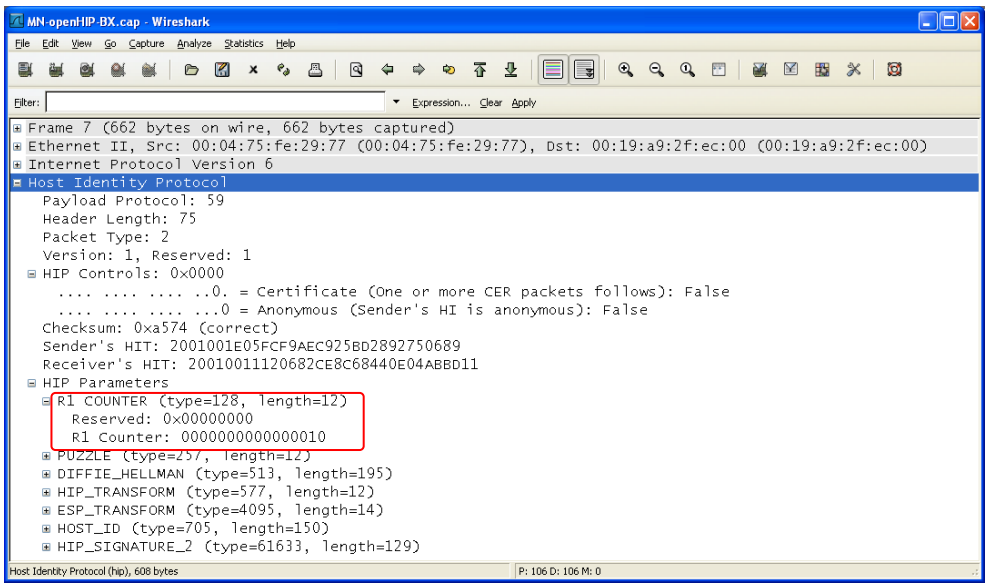

Ilustración 29: detalle del paquete R1 con OpenHIP.

Si hacemos una captura de tráfico vemos que todo es muy parecido a HIPL. La única novedad destacable es que OpenHIP incorpora el parámetro *R1_COUNTER*, que da soporte al mecanismo de seguridad del contador de generaciones de R1.

Por último, cabe destacar que OpenHIP incorpora una interesante solución para la resolución de nombres DNS que permite conectar con un nodo HIP sin conocimiento previo de su dirección IP. Desafortunadamente, esta funcionalidad sólo está disponible en la versión de espacio de usuario y sólo para LSI, es decir, para aplicaciones IPv4. De forma resumida, el funcionamiento es el siguiente:

1. Las aplicaciones que deseen usar HIP deben añadir el sufijo ".hip" al nombre DNS del corresponsal. El demonio hip, que implementa en cierto modo un servidor DNS, intercepta las peticiones de resolución e intenta devolver un HIT o una IP según se use o no dicho sufijo.
2. La dirección IP la obtiene del fichero "know_hosts_identities.xml". En caso de no encontrarla hace una búsqueda DNS adicional del nombre sin

el sufijo ".hip". Si la dirección obtenida es IPv6 se establecerá un túnel ESP IPv6 por el que circulará el trafico IPv4.

3. El HIT y el LSI los obtiene también del fichero "know_hosts_identities.xml". En caso de no encontrarlos puede intentar el intercambio en modo oportunista.

En la siguiente ilustración se muestran unos ejemplos de resolución. Se puede ver que la dirección del servidor DNS es el propio LSI del nodo. Según se incorpore o no el sufijo ".hip", la resolución devuelve un LSI o una IP respectivamente.

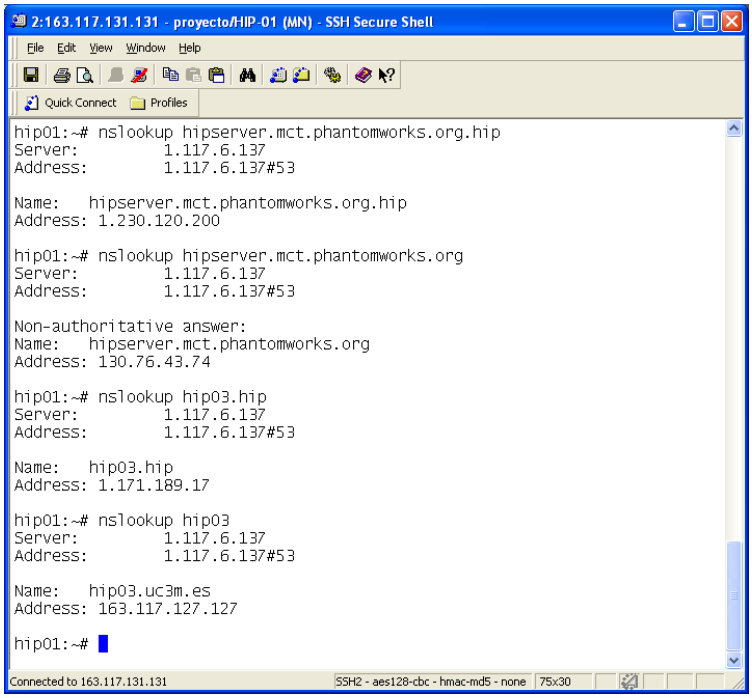

Ilustración 30: ejemplos de resolución de nombres con OpenHIP.

HIP4INTER

En esta implementación, el HI se genera durante la instalación y se almacena en el directorio "/etc/hip", pero pueden generarse otros nuevos con la herramienta "*hip-keygen*". En el nodo corresponsal, que hace el papel de contestador, arrancamos el demonio "hipd" utilizando el *script* "/etc/rc.d/hipd start" (si queremos ejecutarlo en primer plano con mensajes de traza podemos ejecutarlo directamente de la forma "hipd –nl 65535"). En primer lugar, este demonio examina en directorio "/etc/hip" en busca del HI local y, a

continuación, introduce una serie de cambios en la tabla de interfaces y la tabla de rutas local, que detallamos a continuación.

Por un lado, añade un nuevo interfaz virtual llamado “lo1”, al que asigna como dirección IPv6 el HIT correspondiente al HI del nodo. Y por otro, añade las rutas necesarias para que el tráfico correspondiente al prefijo 2001:10::/28 (el espacio de direccionamiento HIP) se curse a través del interfaz “lo1”.

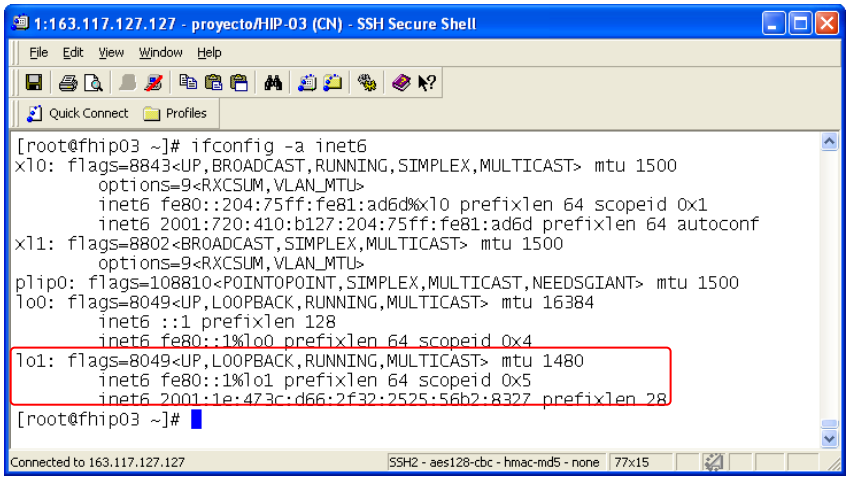

Ilustración 31: tabla de direcciones con HIP4INTER.

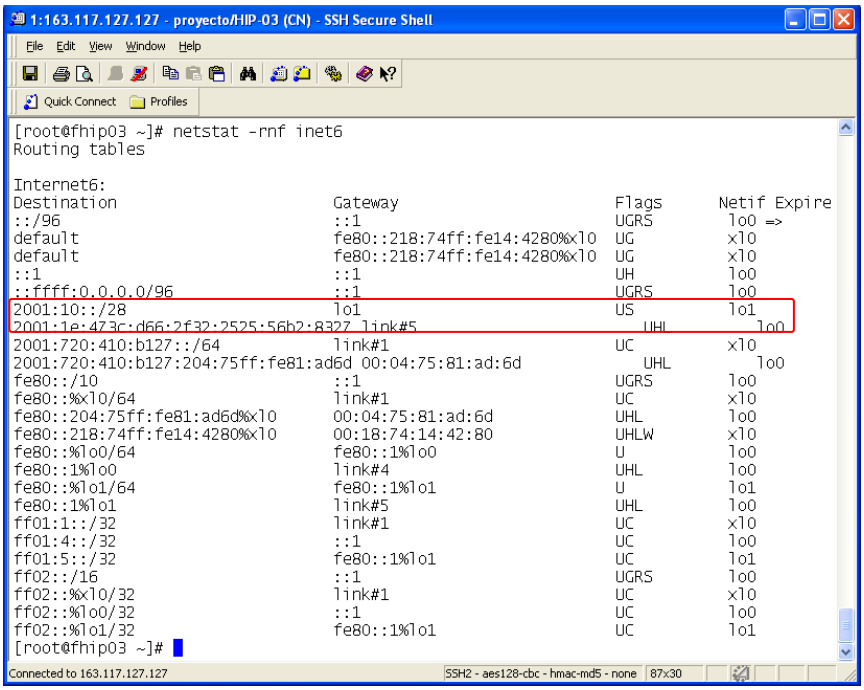

Ilustración 32: tabla de rutas con HIP4INTER.

Situándonos en el nodo móvil, que hace el papel de iniciador, arrancamos el demonio “hipd” y, mediante un *ping*, generamos tráfico hacia el HIT del nodo móvil. Para la resolución del HIT del contestador en una IP inicial, hay que editar previamente el fichero “/etc/hosts” añadiendo dos instancias para cada nodo, una con su IP y otra con su HIT. Esta implementación incluye una modificación en las librerías de resolución de nombres de modo que, cuando el demonio “hipd” está activado, éstas devuelven a las aplicaciones HIT o LSI si están disponibles. Esto hace posible el uso de nombres de una forma muy sencilla.

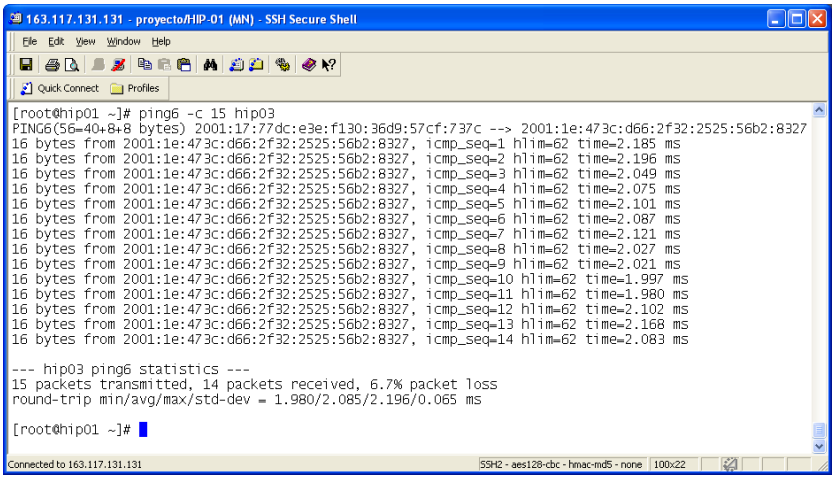

Ilustración 33: comando *ping6* usando un HIT en lugar de IP con HIP4INTER.

Podemos observar que se ha perdido el primer paquete (con icmp_seq=0), y después de repetir la prueba varias veces constatamos que siempre sucede lo mismo. A diferencia de HIPL y OpenHIP, esta implementación no incorpora un mecanismo que almacene el tráfico hasta que la asociación HIP correspondiente esté ya establecida y, por tanto, el paquete que dispara el BEX siempre se pierde. También podemos ver que, a pesar de usar una arquitectura en espacio de *kernel,* obtenemos un RTT mucho más alto que el de los dos casos anteriores. Estos detalles nos hacen suponer que el código de esta implementación se encuentra aún poco optimizado.

Como en el caso de HIPL, al reutilizar la implementación *kernel* de IPsec para cursar el trafico HIP, podemos examinar el estado de las tablas SAD y SPD de IPsec para ver las entradas añadidas por el demonio HIP. Para ello utilizamos el comando “setkey”, disponible para *FreeBSD*, que ha sido modificado de modo que reconoce y decodifica correctamente la asociaciones en modo BEET.

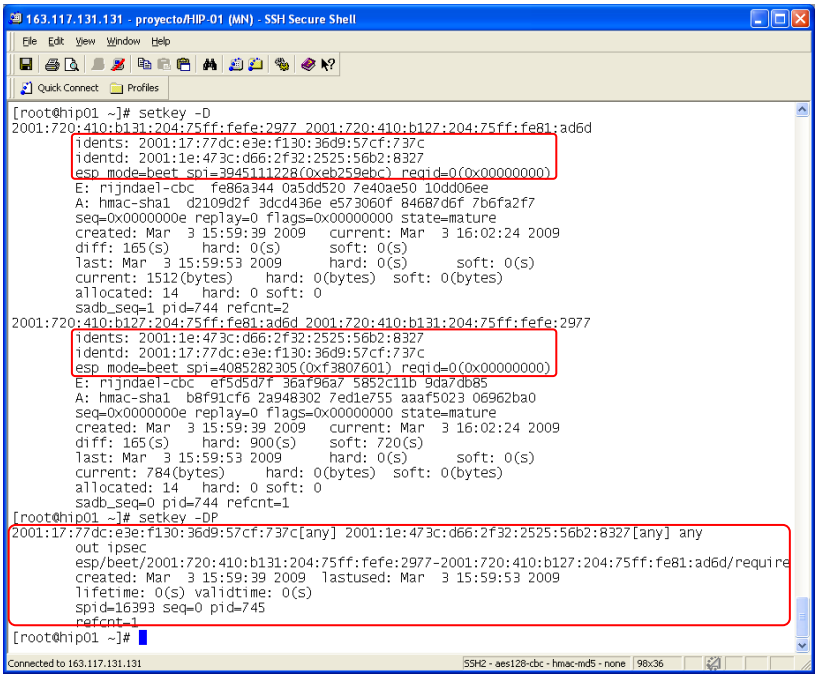

Ilustración 34: tablas SAD y SPD tras el Intercambio Básico con HIP4INTER.

INTERCAMBIO BÁSICO CON RVS

De las tres implantaciones de HIP disponibles, sólo HIPL y OpenHIP parecen incluir las especificaciones de la RFC 5204 sobre el servidor *randezvous*. Con OpenHIP sólo fue posible conseguir un funcionamiento correcto con IPv4. Con IPv6 aparecía un error grave en el código (*segmentation fault*) relacionado con el sistema de registro que provocaba la cancelación del proceso “hipd”. Este problema pese a ser conocido por los desarrolladores aún no parece haber sido solucionado. Con HIPL, sin embargo, se pudo realizar con éxito un intercambio básico a través de RVS, el cual detallamos a continuación.

Para conseguir que el iniciador comience el intercambio básico a través de un servidor RVS, basta con sustituir en el fichero “/etc/hosts” la IP inicial del contestador por la del propio RVS. Además, el servidor RVS debe ser iniciado utilizando el comando “hipconf add service rvs”, que indica al demonio “hipd” que deseamos que ofrezca este servicio a otros nodos HIP. Existe un fichero “/etc/hip/relay_config” en el que se pueden configurar algunos parámetros del servidor RVS, así como una lista blanca para enumerar aquellos nodos a los que se desee permitir registrarse. Por último, el nodo contestador tiene que registrarse en su RVS para poder recibir nuevas conexiones a través suyo. Esto se realiza con el comando “hipconf add server rvs <RVS-HIT> <RVS-IP>”.

En la captura de tráfico de la siguiente ilustración, hecha desde el propio servidor RVS, podemos ver todo el proceso. Primero, vemos como el iniciador envía paquetes I1 al RVS sin tener ningún efecto. Esto se debe a que el HIT del contestador todavía no se ha registrado en el servidor. A continuación, se produce el intercambio básico entre contestador y RVS que lleva implícito el proceso de registro. A partir de ese momento, el siguiente I1 del iniciador se reenvía al contestador continuado después el intercambio básico de forma directa sin implicar al RVS. También podemos ver los parámetros FROM y RVS_HMAC que el servidor incluye en el paquete I1 al reenviarlo[42].

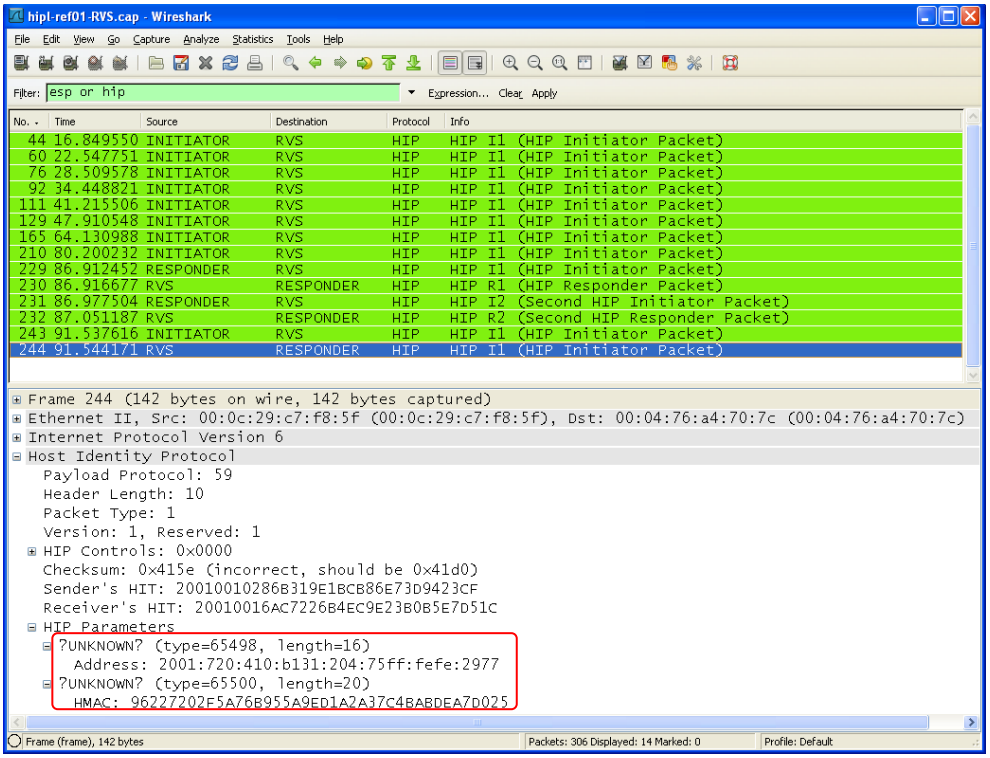

Ilustración 35: proceso de registro y posterior redirección del I1.

A continuación, vemos otra captura tomada esta vez desde el iniciador. En ella puede verse como, después de que el servidor RVS haya reenviado el paquete I1, el intercambio básico continúa entre iniciador y contestador estableciéndose el consiguiente túnel ESP. También podemos ver como el paquete R1 incluye el parámetro VIA_RVS, cuya misión es la de infirmar al iniciador de que su paquete I1 ha sido modificado y reenviado a través de un RVS, cuya dirección IP está indicada.

[42] Los parámetros correspondientes a la operativa del servidor RVS no están totalmente decodificados, por lo que aparecen como *unknown*. El analizador es una versión parcheada de *Wireshark* que incluye el protocolo HIP aunque, como vemos, de forma parcial.

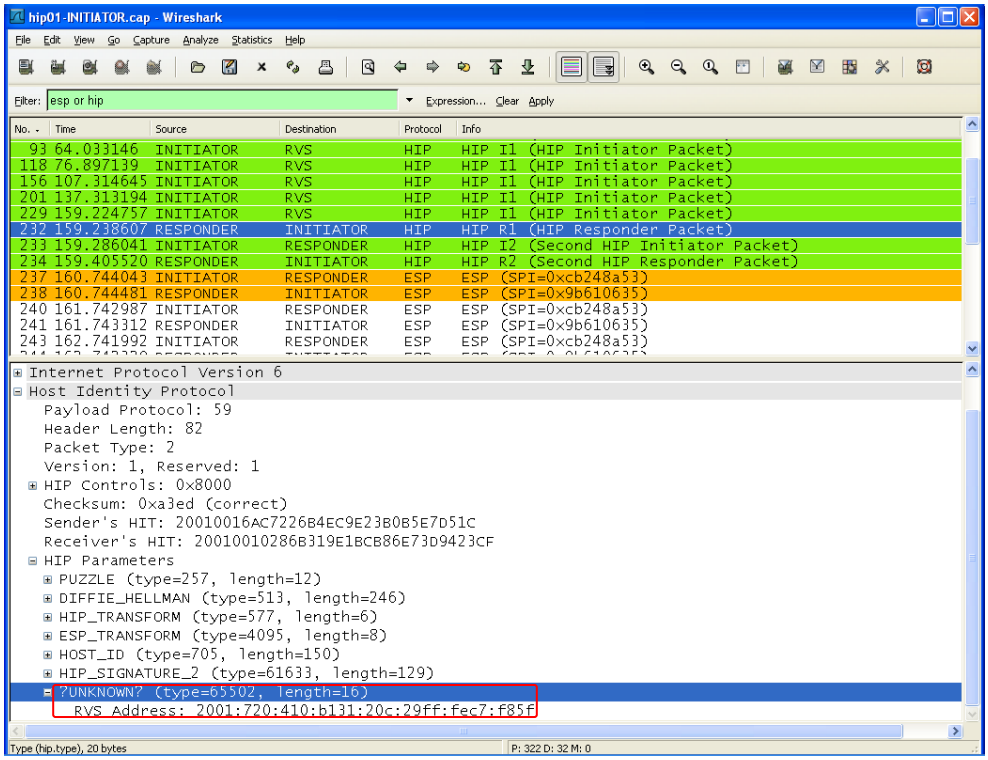

Ilustración 36: BEX a través de un RVS visto desde el iniciador.

INTEROPERATIVIDAD

En el marco del grupo de trabajo del IETF, los desarrolladores suelen realizar pruebas de interoperatividad para asegurar la validez del código de sus implementaciones. Para facilitar este tipo de pruebas asimismo existen nodos de referencia accesibles por Internet con las últimas versiones estables de cada implementación.

A lo largo de nuestro trabajo también realizamos algunas pruebas de interoperatividad, pudiendo comprobar su correcto funcionamiento con una sola excepción. Utilizando HIPL como iniciador y HIP4INTER como contestador el BEX funcionaba con éxito, pero al intercambiar los papales HIP4INTER generaba un error en el procesamiento del R1 y se interrumpía el intercambio. Se informó de esta anomalía a los desarrolladores por medio de su dirección de correo electrónico de contacto y el problema fue resuelto rápidamente[43]. Se trataba, efectivamente, de un error en la decodificación de uno de los parámetros Diffie-Hellman del paquete R1.

[43] Esta corrección quedó corregida en el *snapshot-2007.12.26*.

MOVILIDAD CON DISTINTAS APLICACIONES

En la documentación disponible para las tres implementaciones se describen pruebas de movilidad elementales que, invariablemente, consisten en añadir manualmente una nueva dirección IP a uno de los nodos de una asociación HIP previamente establecida y, a continuación, borrar la dirección IP antigua también manualmente. El software de HIP detecta la desaparición de la IP que venía usando hasta ese momento, busca una dirección alternativa e intenta usarla inmediatamente arrancando el proceso de actualización de la asociación. Este tipo de simulación de movimiento tiene por objetivo probar el funcionamiento del método de actualización del protocolo HIP, pero evitando la mayoría de los problemas prácticos que implica la movilidad real de un nodo. Nuestra pretensión era la de realizar pruebas de movimiento algo más realistas que incluyeran un cambio de subred IP. Es decir, un cambio en el nivel de enlace que suponga la necesidad de reconfigurar el nivel de red, con la exigencia añadida de que todo el proceso de re-direccionamiento se realizase de forma autónoma.

En nuestro caso concreto, para conseguir este tipo de movimientos fue necesario entrar en consideraciones sobre dos aspectos que no se habían cubierto en pruebas anteriores:

- La configuración adecuada de ciertos parámetros del protocolo ND (*Neighbor Discovery*) de IPv6 (41) en los *routers* de acceso, como son, la frecuencia de los anuncios y el ajuste en los tiempos de validez de la dirección IP y de las rutas local y por defecto.
- La disponibilidad de una gestión local de direcciones y rutas en el nodo móvil. Es decir, un sistema de detección de movimientos asociado a un algoritmo de decisión para la reconfiguración IP (ver el apartado "Direccionamiento y detección del movimiento"). Ninguna de las actuales implementaciones de HIP cuenta con un sistema de este tipo mientras que MIPL[44], por el contrario, sí lo tiene.

En un primer momento, el desarrollo de las pruebas se dirigió a experimentar con los diferentes parámetros de configuración del protocolo ND. La primera medida para intentar conseguir los objetivos fue aumentar la frecuencia de los

[44] MIPL es una implementación de Mobile IPv6 para *Linux* (http://www.mobile-ipv6.org/) que utilizaremos más adelante para las pruebas comparativas entre HIP y MIP (ver apéndice de instalación de elementos auxiliares).

anuncios RA, punto fundamental para que el nodo móvil detecte lo antes posible la disponibilidad de una nueva configuración IP válida.
Un *router* envía sus anuncios RA separados por un tiempo aleatorio, elegido dentro un intervalo definido por unos tiempos máximo y mínimo entre anuncios[45]. En adelante, para referirnos a estos valores usaremos la notación [máximo, mínimo]. Para estos dos parámetros las especificaciones del protocolo ND establecen unos valores por defecto de [600,200] segundos, y unos valores mínimos de [4,3] segundos; esto nos da un tiempo medio entre anuncios de 3.5 segundos como mínimo. Estos tiempos, más bien altos, se justifican porque los anuncios RA, o la falta de ellos, no fueron concebidos para detectar fallos de conectividad; para este cometido hay estipulado un algoritmo conocido como *Neighbor Unreachability Detection (NUD)*. Pero por otra parte, las especificaciones de *Mobile IP* (5) reconocen que esta limitación puede ser inapropiada si se quiere facilitar a los nodos móviles la oportuna capacidad de detección de movimiento. Así, aquellos *routers* que previsiblemente vayan a dar servicio a nodos móviles podrían rebajan, de forma excepcional, los valores mínimos del intervalo a [70,30] milisegundos, es decir, un tiempo medio mínimo entre anuncios de 50 milisegundos[46]. Sin embargo, también se advierte que usar estos valores mínimos en determinadas redes con bajo ancho de banda, particularmente redes inalámbricas, puede causar degradación del servicio y otros problemas.

En las primeras pruebas, para los anuncios RA se utilizó un intervalo de [3,1] segundos, recomendado en la documentación de MIPL y que daba buenos resultados para éste software. Sin embargo, ninguna de las implementaciones disponibles de HIP era capaz de restablecer las conexiones de forma satisfactoria. A raíz de estas primeras pruebas se puso de manifiesto que la disponibilidad de una nueva configuración IP no era suficiente para recuperar la capacidad de enviar paquetes con éxito y el problema se encontraba en la gestión de la tabla de rutas del nodo móvil. Tras obtener una nueva configuración IP, esta tabla seguía conteniendo entradas para las rutas local (*local-link*) y por defecto correspondientes a la configuración IP anterior que tardaban un tiempo determinado en descartarse. Esto significa que durante

[45] Parámetros “MaxRtrAdvInterval” y “MinRtrAdvInterval”, (ver apéndice de configuración de los *routers*).
[46] Esta posibilidad está contemplada de forma distinta en las dos implementaciones del protocolo ND que se han probado. La implementación de software libre para *Linux*, *radvd*, dispone del parámetro de configuración “AvdIntervalOpt on” que nos permite bajar los tiempos hasta los límites citados en las especificaciones de Mobile IPv6. En el caso de los *router Cisco* no existe tal opción y el límite mínimo para el tiempo medio entre anuncios es de 500ms (ver apéndice de configuración de los *routers*).

dicho tiempo los paquetes del nodo móvil se enviaban con la IP correcta, pero se encaminaban de forma errónea:

- Los paquetes dirigidos a la antigua red local, se enviaban a través de la antigua ruta local, en lugar de usar la nueva ruta por defecto.
- Los paquetes dirigidos a cualquier otra red usaban la antigua ruta por defecto.

Como resultado, en la mayoría de los casos los intercambios de actualización se perdían y las asociaciones HIP quedaban rotas, y en caso de conseguir restablecerse tardaban un tiempo mucho más largo del que podía observarse con MIPL en el mismo escenario.

Las pruebas continuaron intentando encontrar una configuración de los anuncios RA que permitiera superar este problema. El intervalo entre anuncios también controla el tiempo de validez del *router* anunciado, y por lo tanto, de la ruta por defecto del nodo móvil. Este tiempo es configurable[47], aunque tiene como límite mínimo el tiempo máximo entre anuncios y, por defecto, se establece como el triple de éste. También se pudieron mejorar los resultados reduciendo el tiempo de validez del prefijo anunciado[48]. Este parámetro controla los tiempos de validez tanto de la dirección IP asociada al prefijo, como de su ruta local. Este tiempo de validez no está limitado por el tiempo entre anuncios, pero si es más bajo que éste la dirección IP aparecerá y desaparecerá de la tabla de direcciones provocado al nodo desconexiones erráticas. Por otra parte, las entradas de la tabla de rutas no se descartan inmediatamente cuando expira su tiempo de validez. A menudo, tras la expiración, se ejecuta el procedimiento NUD, lo que añade unos 12 segundos más antes de que dicha ruta quede finalmente descartada.

Reduciendo al mínimo todos los parámetros mencionados, los resultados, aunque un poco mejores, seguían siendo irregulares y poco claros, dependiendo de factores diversos que no siempre fuimos capaces de aclarar. Veamos algunos ejemplos:

- Tiempo de conmutación: es decir el tiempo transcurrido desde que se abandona una red hasta que se conecta a la siguiente. En una conmutación rápida, el nodo móvil pasa de una red a otra de forma casi instantánea, y la actualización falla inicialmente porque el paquete de actualización se envía a través del las rutas antiguas que aún no han

[47] Parámetro “AdvDefaultLifetime” (ver apéndice de configuración de los *routers*).
[48] Parámetro “AdvValidLifetime” (ver apéndice de configuración de los *routers*).

caducado. Las posteriores retransmisiones pueden conseguir que la actualización llegue finalmente a su destino, aunque con un retraso considerable. En una prueba de conmutación lenta el nodo móvil se desconecta durante 15 ó 20 segundos y después se conecta a una nueva red, dando tiempo a que las rutas antiguas sean descartadas antes de iniciar el intercambio de actualización. Paradójicamente, el resultado era mejor con una conmutación lenta, lo que implicaba que el mecanismo de retransmisiones implementado no siempre funcionaba como era de esparar.

- No todas las implementaciones del protocolo ND son exactamente iguales. En nuestras pruebas utilizamos *routers Cisco* y *Linux* que, para valores iguales de los parámetros de configuración, daban resultados diferentes sobre las rutas del nodo móvil. Las rutas derivadas de anuncios procedentes de un *router Cisco* ejecutaban siempre el procedimiento NUD antes de descartarse. En el caso de *routers Linux,* el procedimiento NUD sólo se ejecuta para la ruta local, lo que implica que la ruta por defecto se descarta con mayor rapidez. Así, cuando el nodo móvil visita la red de su corresponsal y se mueve después a otra red cualquiera, la actualización de la asociación tiene siempre peores resultados. Para otros movimientos los *router Linux* dan mejor resultado que los *Cisco*.
- La implementación HIPL presentaba problemas debido a que realizaba una sola retransmisión del paquete inicial del intercambio de actualización; insuficiente cuando las rutas tardan un tiempo más largo en desecharse. A través de su lista de correo para usuarios se informó a los implementadores de dicho problema, que fue rápidamente corregido con un mayor número de retransmisiones[49].

En este punto, se hizo evidente que no era posible conseguir nuestros objetivos sin contar con una gestión adecuada de las tablas de direcciones y rutas del nodo móvil. Era necesario implementar un procedimiento de detección de movimiento que disparase el borrado de las direcciones y rutas antiguas cuando se dispusiera de otras nuevas.

Para conseguir esto de una forma sencilla se propuso utilizar un pequeño programa independiente que corriese de forma concurrente con el software de HIP. Este programa-herramienta debería escuchar los paquetes RA en la red, guardando la información del *router* y el prefijo anunciados. La recepción de un RA con información distinta sería considerada prueba suficiente de que se ha

[49] Esta corrección del código se encuentra reflejada en el parche 279, publicado el 29-09-2007.

producido un movimiento[50]. Aunque, evidentemente, el problema de la detección de movimiento es algo mucho más complejo, dentro de nuestro escenario de pruebas este tratamiento sencillo resulta suficientemente eficaz (ver el capítulo de detección de movimiento). Una vez detectado el movimiento, se puede realizar el borrado de las entradas correspondientes en las tablas de direcciones y rutas mediante llamadas a comandos del SO.

Por fin, con este método, se consiguió un comportamiento correcto y estable en los movimientos con HIP, equiparable al obtenido con MIPL. De las tres implementaciones de HIP, HIPL y HIP4INTER dieron buenos resultados, sin embargo, la implementación OPENHIP no logro funcionar con la debida estabilidad. A través de la lista de correo para usuarios de OPENHIP, se informó del problema a los implementadores. Estos nos ofrecieron alguna orientación sobre cuál podría ser el origen del fallo (probablemente la selección de la dirección IP origen) pero hasta la fecha el problema persiste.

A continuación, utilizando la herramienta descrita se realizaron pruebas de movilidad con distintas aplicaciones. La única exigencia para probar una determinada aplicación era que soportase IPv6 de forma nativa, de modo que se pudieran utilizar HIT directamente como si fueran direcciones IPv6. Las pruebas descritas a continuación se realizaron únicamente con la implementación HIPL en espacio de *kernel* sobre un sistema *Linux*/*Debian*-*etch*. El motivo es que dicha implementación es la que nos dio los mejores resultados en las pruebas anteriores, junto con una mayor familiaridad en el uso de *Linux* sobre otras opciones disponibles.

SECURE SHELL - SSH

Como servidor y cliente SSH se utilizó la implementación de ***OpenSSH***[51], una versión libre de las herramientas de comunicación SSH desarrollada por el *OpenBSD Project*. Su instalación se realiza con los paquetes *openssh-client* y *openssh-server* en los nodos cliente y servidor respectivamente. Por defecto,

[50] Como base para este programa se utilizó la herramienta *radvdump,* contenida en el paquete *radvd* para *Linux*. Dicha herramienta escucha, decodifica e imprime en pantalla los anuncios RA. De este modo, sólo fueron necesarios algunos cambios para incluir la lógica de detección de movimiento y borrado de configuraciones IP antiguas. El parche para la modificación de esta herramienta fue remitido a lista de correo de la implementación HIPL, donde tuvieron a bien incluirlo en su árbol de fuentes como colaboración. Parche 359, publicado el 3-10-2008 (ver apéndices).

[51] http://www.openssh.com/

tanto cliente como servidor, soportan y aceptan conexiones sobre IPv6. De este modo, arrancado el servidor normalmente, en el cliente ejecutaremos:

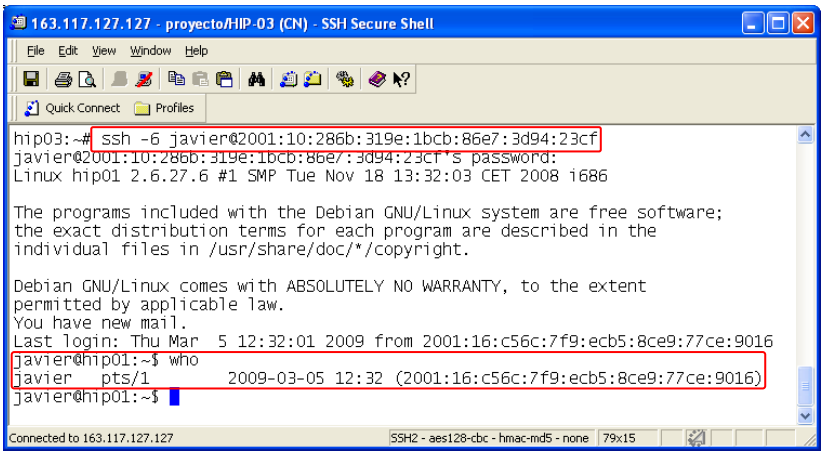

Ilustración 37: sesión SSH usando un HIT en lugar de IP.

Las pruebas de movilidad funcionaron correctamente con tiempos de conmutación cortos y largos, reanudándose la sesión pocos segundos después de la re-conexión y de forma reiterada tras múltiples saltos. Sólo en el caso de tiempos de conmutación considerables (al menos un minuto), o de varios saltos demasiado rápidos la sesión quedó bloqueada, aunque no en todas las ocasiones. Tales fallos fueron más achacables a la propia aplicación o a los temporizadores TCP que a HIP, ya que, con un simple *ping* se pudo comprobar que seguía existiendo conectividad entre el nodo móvil y su corresponsal.

TRANSFERENCIA DE FICHEROS - FTP

Atendiendo a la necesidad de soportar IPv6, utilizamos ***VSFTPD***[52] como servidor y ***LFTP*** [53] como cliente. Ambas son implementaciones de *software* libre bajo licencia GPL y se instalan mediante un paquete con su mismo nombre. En el servidor sólo es necesario activar el soporte IPv6[54], que esta desactivado por defecto, y arrancar normalmente el servicio. A continuación, el cliente hace una conexión al HIT del servidor:

[52] http://vsftpd.beasts.org/
[53] http://lftp.yar.ru/
[54] El fichero "/etc/vsftpd.conf" deberá contener las siguientes opciones: "*listen=NO*" y "*listen_ipv6=YES*".

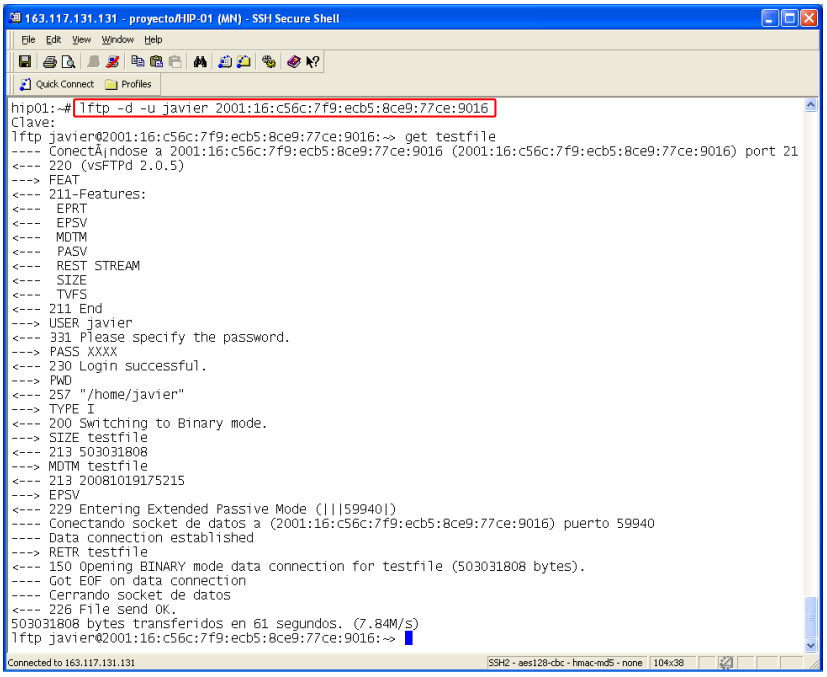

Ilustración 38: sesión FTP usando un HIT en lugar de IP.

Las pruebas de movilidad funcionaron correctamente usando el cliente ftp tanto el modo activo como el pasivo[55]. Como en el caso de SSH se observaron algunos bloqueos de la aplicación tras largos periodos de desconexión o varios movimientos rápidos, aunque de igual forma éstos eran más achacables a la aplicación o los temporizadores TCP que al propio protocolo HIP.

Se realizaron, además, medidas de la velocidad de transferencia con una conexión HIP, comparando sus resultados con una conexión IPv6 simple y un túnel ESP modo transporte[56]. Para obtener estos resultados se realizaron diez medidas del tiempo de transferencia de un fichero de 500 Mbytes obteniendo la media para cada uno de los tres escenarios. Se pudo observar que, frente a la conexión IPv6 simple, la conexión HIP mostraba una pérdida de rendimiento del 17%. Por otro lado, la conexión a través del túnel ESP mostraba una pérdida de rendimiento del 11%; un valor similar, aunque algo menor. Teniendo en cuenta que el tráfico de datos, una vez realizado el intercambio básico, se procesa en ambos casos a través del código IPsec incluido en el *kernel*, y que la única diferencia entre ambos túneles es su modo de trabajo (modo transporte o modo BEET), la diferencia de rendimiento observada sería achacable a la

[55] Por defecto *lftp* trabaja en modo pasivo. Para activar el modo activo se debe incluir en el fichero "*$HOME/.lftprc"* el comando "*set ftp:passive-mode off"*.

[56] Para realizar esta prueba los túneles se establecieron manualmente mediante la herramienta "*ip xfrm"* incluida en el paquete *iproute2.* Para conseguir resultados comparables, se utilizaron algoritmos de cifrado y verificación (aes y sha1) y longitudes de clave idénticas a las de los túneles creados por HIPL.

implementación del modo BEET; quizá más compleja o no lo bastante optimizada.

En la siguiente ilustración puede verse un gráfico comparativo de las velocidades medias de transferencia obtenidas. Los resultados detallados de la prueba pueden verse en el apéndice correspondiente.

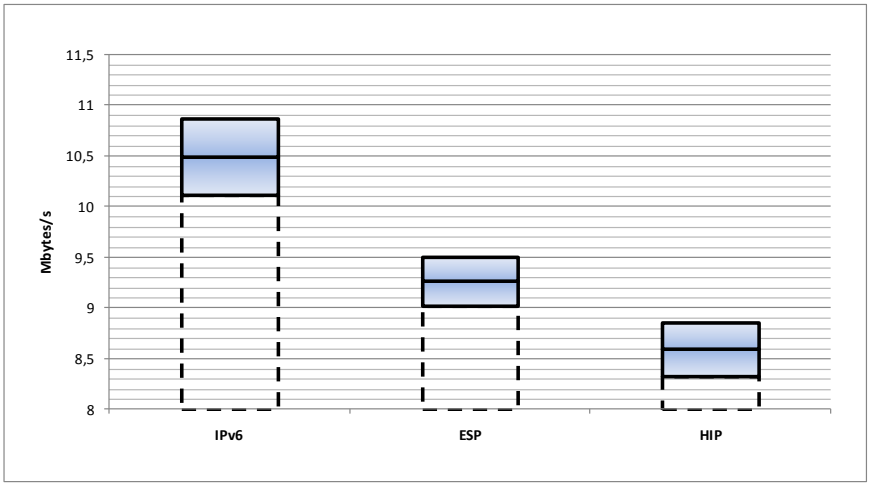

Ilustración 39: velocidad de transferencia FTP (Intervalo de Confianza 95%).

VIDEO *STREAMING* - VLC

Para las pruebas de *streaming* se utilizó, tanto en el servidor como el cliente, ***VLC media player***[57], una implementación de *software* libre con licencia GNU disponible para numerosas plataformas y con soporte IPv6. VLC puede usar TCP o UDP como protocolos de transporte y se realizaron pruebas de ambos con diferentes resultados.

Con TCP/HTTP la secuencia es la siguiente: primero, en el servidor arrancamos un *listener* en el puerto 8080, que esperará solicitudes de conexión a la sesión de *streaming*. El comando es el siguiente:

```
vlc <videofile> --ipv6 --sout '#std{access=http,mux=ts,dst=:8080}'
```

En el anterior comando se podría haber incluido el HIT del cliente de modo que la sesión estuviese restringida a éste, sin embargo, VLC parece presentar un error en el *parser* de su línea de comandos y es incapaz, en este caso, de separar correctamente la dirección IPv6 (o el HIT) del puerto que se pone a continuación.

[57] http://www.videolan.org/vlc/

Después, conectaremos el cliente a la sesión de *streaming* mediante el siguiente comando, que utiliza un HIT como dirección:

```
vlc http://[<HIT de servidor>]:8080
```

A continuación, vemos la imagen de una sesión de *streaming* donde puede observase (línea inferior) el HIT y el puerto desde donde el reproductor está recibiendo los datos.

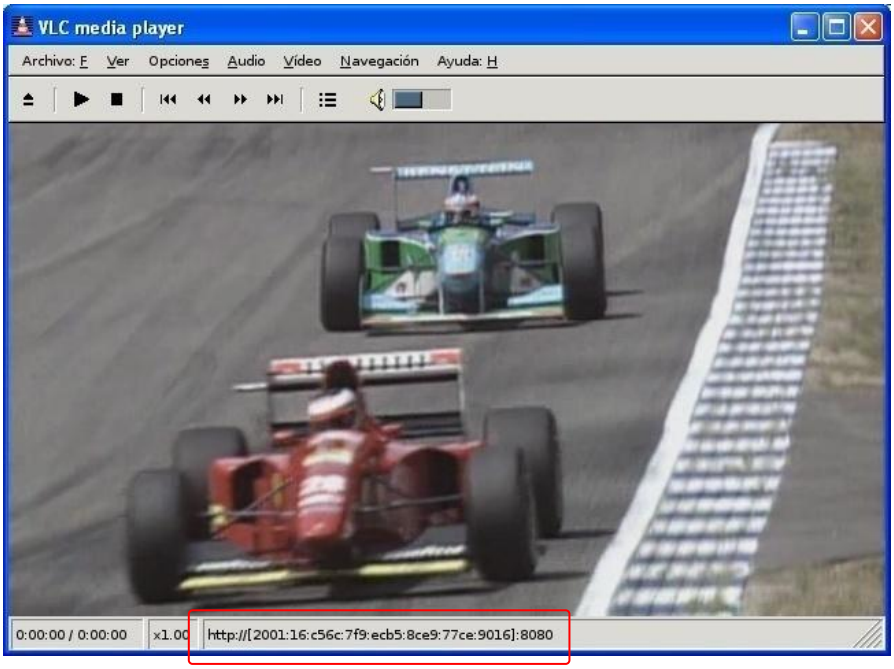

Ilustración 40: *streaming* TCP/HTTP con VLC usando un HIT en lugar de IP.

De forma similar, también se hicieron pruebas sobre RTP/UDP siguiendo la siguiente secuencia. Primero, en el servidor arrancamos la sesión con el comando:

```
vlc -vvv <any avi video file> --ipv6 --sout '#std{access=rtp,mux=ts,dst=[<HIT del cliente>]:1234}'
```

A partir de ese momento el servidor empieza a enviar paquetes al cliente, el cual para recibirlos deberá ejecutar lo siguiente:

```
vlc –vvv udp:
```

Con este comando, el cliente comenzará a reproducir cualquier sesión de *streaming* que reciba por UDP. Desafortunadamente, este método de

streaming provocaba que a los pocos segundos de iniciarse la sesión el servidor se bloqueara completamente, teniendo que recurrir al reinicio hardware como única solución.

Se informó de este problema a los implementadores de HIPL mediante su lista de correo, adjudicándole un *bug report* que fue finalmente resuelto al cabo de algunos meses[58]. Aparentemente el problema estaba relacionado con el código de ESP modo BEET.

COMPARACIÓN CON MIPL

Como última prueba de este trabajo, se propuso hacer una comparativa cuantitativa de la eficacia del procedimiento de traspaso del protocolo HIP frente al protocolo MIP. Para ello se diseño una prueba que permitiera medir el tiempo de desconexión provocado por un movimiento y se repitió dicha prueba en condiciones idénticas para ambos protocolos corriendo sobre los mismos nodos. La prueba básica consiste en enviar entre el nodo móvil y su corresponsal un tren de 200 paquetes de 1Kb de tamaño a lo largo de un tiempo de 20s, es decir, separados entre sí por 100ms. Simultáneamente, se realiza un movimiento de modo que el proceso de traspaso empiece y termine dentro de los 20s de la prueba. De esta forma, contabilizando el número de paquetes perdidos se puede hacer una estimación del tiempo de desconexión del nodo móvil. Esta prueba básica se repite a su vez en 20 ocasiones para obtener una medida más fiable en términos estadísticos.

MIP puede usarse con cuatro posibles configuraciones, según se utilicé *eager switching* (ES) o *lazy switching* (LS)[59] combinado con la opción de *route optimization* (RO) activada o no. Para asegurarnos de que obteníamos el mejor resultado posible del protocolo MIP, se realizaron pruebas con todas las combinaciones de estos dos parámetros de su configuración. Debido a las limitaciones de HIPL ya descritas anteriormente, éste sólo fue probado con *eager switching*. Por último, todas estas pruebas se repitieron con tiempos medios entre anuncios RA de 60ms., 1s. y 3s respectivamente

[58] Esta corrección del código se encuentra reflejada en el parche 359, publicado el 30-10-2008.
[59] A partir del análisis de capturas de tráfico, aparentemente, la diferencia radica en que con LS el nodo móvil envía 2 ó 3 paquetes *Neighbor Solicitation* antes de enviar su paquete *Binding Update* (BU), independientemente de la llegada de anuncios RA nuevos. Por el contrario, con ES el BU se envía inmediatamente al recibirse un nuevo RA.

Los resultados obtenidos reflejaron tiempos de traspaso muy similares entre HIP y MIP con ES, independientemente de si se utilizaba RO o no. Además, como era previsible, cuando se usaba MIP con LS los resultados eran aproximadamente un segundo mayores que los de HIP y los de MIP con ES. En las siguientes ilustraciones pueden verse unos gráficos comparativos de los tiempos medios de traspaso obtenidos con un intervalo de confianza del 95%. Los resultados detallados de la prueba pueden verse en el apéndice correspondiente.

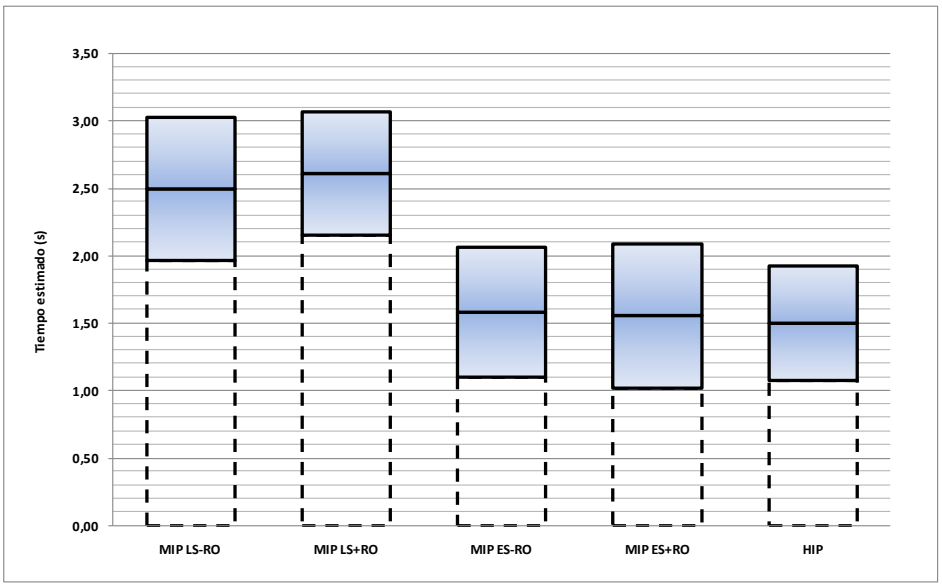

Ilustración 41: tiempo de traspaso con un RA cada 60ms (I.C. 95%).

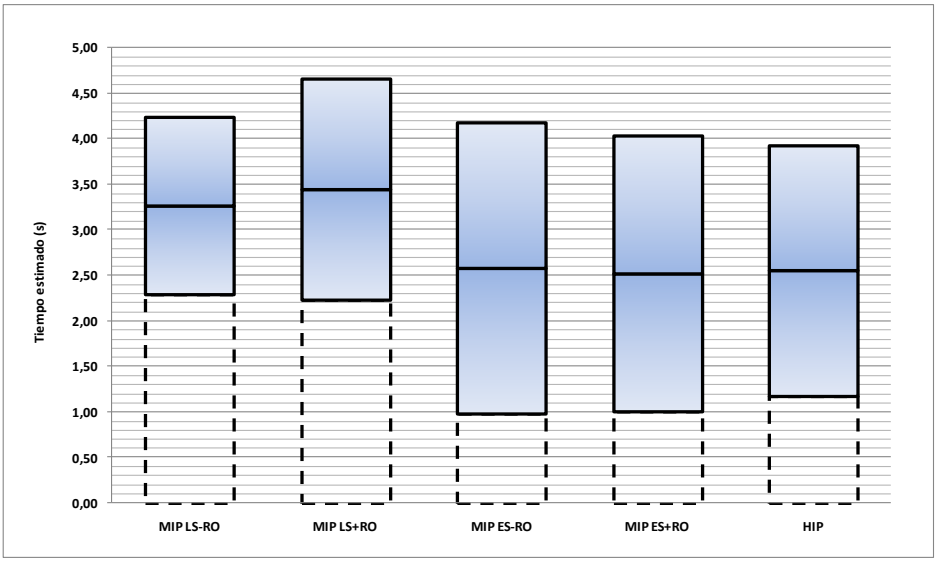

Ilustración 42: tiempo de traspaso con un RA cada 1s (I.C. 95%).

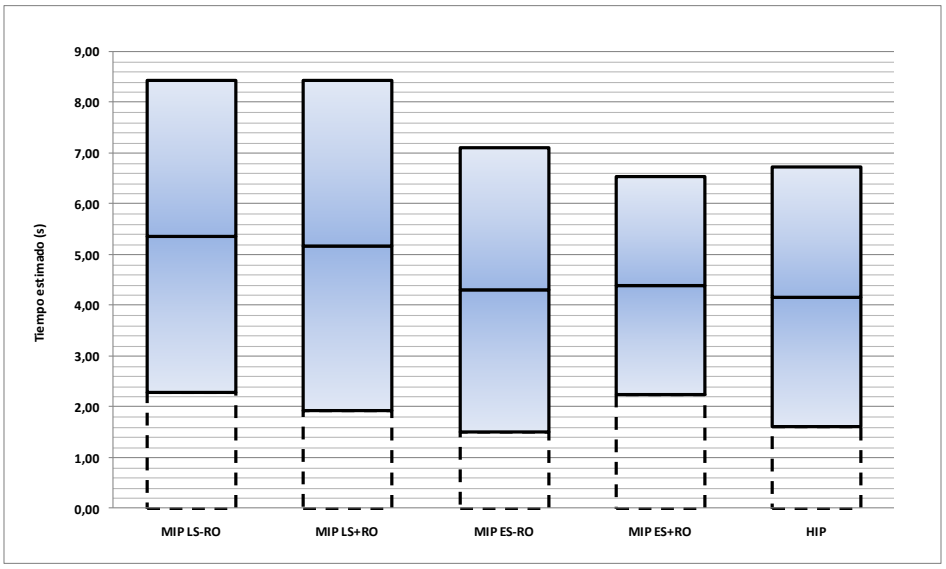

Ilustración 43: tiempo de traspaso con un RA cada 3s (I.C. 95%).

PARTE IV: CONCLUSIONES

Actualmente, existe cierta necesidad de extender la semántica del modelo TCP/IP para conseguir dotar a Internet de nuevas funcionalidades, como la movilidad o el *mutihoming*, o recuperar otras, como la conectividad punto a punto de forma sencilla y directa, perdida por la existencia de ciertos dispositivos intermedios (*middleboxes*). En esta área, hay numerosas iniciativas que se centran en la solución de algunos de estos problemas concretos, sin embargo, HIP propone un enfoque más fundamental que puede aplicarse de forma general. De ahí también su mayor desventaja, ya que la adopción de HIP supondría un despliegue en todos los nodos comunicantes y la introducción de nuevos elementos de infraestructura (Servidores RVS/*Relay*), así como la posible modificación de otros (servidores DNS).

De las tres implementaciones disponibles de HIP, actualmente la más activa es HIPL, aunque todas se mantienen actualizadas con las últimas versiones de las RFC. Sin embargo, se encuentran aún en una fase totalmente experimental lo que hace que la documentación sea bastante básica, su funcionamiento inestable y su utilización poco atractiva. Como ejemplo de lo anterior, cabe destacar que ninguna implementación incluye la función de detección de movimiento, lo que limita considerablemente su experimentación. Por este motivo, las pruebas de este trabajo se han centrado en la validez básica del protocolo HIP, siendo los resultados en este aspecto totalmente favorables.

También, se ha probado factible la movilidad con HIP y aplicaciones IPv6 nativas, sin necesidad de que estas sean adaptadas al nuevo espacio de direccionamiento. Otra cosa diferente es la capacidad de una aplicación para mantener o recuperar la sesión tras periodos de desconexión significativos, aspecto que se ha mostrado problemático y limitante para la movilidad como servicio al usuario.

En comparación con MIP, una propuesta para la movilidad más consolidada, HIP presenta alguna ventaja, como un espacio de direccionamiento totalmente independiente de la topología de red y una integración nativa con el protocolo ESP. Por su parte, MIP tiene como principal ventaja la capacidad de funcionar de modo transparente, es decir, sin introducir ninguna modificación en el nodo corresponsal. Sin embargo, esta ventaja se pierde cuando se usa la extensión *Route Optimizacion* introducida para paliar el enrutamiento triangular; un problema del que HIP carece. Los resultados obtenidos respecto al tiempo de

re-direccionamiento son muy similares para ambos protocolos lo que indica que el concepto adoptado por HIP, la actualización en modo directo es, al menos, tan eficiente como el modo indirecto usado por MIP.

TRABAJO FUTURO

Las especificaciones básicas de HIP están en fase RFC experimental y se está trabajando para proponerlos como estándar. En opinión de algunos de sus promotores dichas especificaciones están prácticamente listas, pero resulta necesaria mayor experiencia, especialmente en despliegues a gran escala. Debido a que HIP introduce modificaciones de calado en la arquitectura TCP/IP, su despliegue podría tener importantes consecuencias en el uso y la operativa de Internet. El ámbito de futuros trabajos es, por tanto, considerablemente amplio.

En un nodo HIP puede ser necesario introducir nuevas herramientas, como por ejemplo, para gestionar las identidades locales de forma segura y asociar cada una de ellas a sus aplicaciones correspondientes. También puede ser necesario modificar el funcionamiento de otras herramientas, como pueden ser las de gestión del *firewall* local para utilizar HIT además, o en lugar, de direcciones IP. Otra futura línea de trabajo podría ser la creación o adaptación de aplicaciones para usar una API HIP nativa, es decir, con acceso directo a los parámetros del protocolo HIP y con capacidad para modificarlos según sus necesidades.

Como ya se ha mencionado anteriormente, es necesario introducir en las implementaciones de HIP un método para la detección de movimiento. En este sentido, los desarrolladores de OpenHIP han liberado recientemente una primera implementación de SHIM6 (40), una iniciativa para el soporte de *multihoming* sobre la base de mensajes de control HIP, incluyendo el protocolo REAP (*REAchability Protocol*) (45) para la detección de pérdidas de conexión y exploración de nuevos rutas de comunicación. Los implementadores de OpenHIP esperan poder reutilizar para HIP los métodos de detección de fallos de SHIM6.

El despliegue de HIP puede tener un impacto importante en determinados elementos de infraestructura de la red, como *firewalls* corporativos y dispositivos NAT o *proxy*. En principio, cualquier tipo de entidad intermedia (*middlebox*) es susceptible de incorporar la capacidad de leer y usar la información de las asociaciones HIP que lo atraviesan, teniendo la posibilidad de realizar sus funciones en base a Identificadores de Host. Sin embargo, esto presenta previsibles dificultades prácticas. Dado que el espacio de nombres HIP es plano y, por lo tanto, no es posible la agregación de identificadores, la configuración y administración de estos dispositivos puede convertirse en algo demasiado prolijo. Actualmente, existen iniciativas para introducir algún tipo de

jerarquía en el espacio de nombres HIP, dividiendo los identificadores en dos partes: un prefijo asignado según su dominio de gestión y un identificador de nodo más corto de estructura plana (42).

El problema de la resolución de nombres en HIT y direcciones IP es un tema todavía bastante abierto. Existen varias aproximaciones: desde el uso de DNS mediante modificaciones, extensiones o *proxies*, hasta el uso de otros mecanismos como DHT (*Distributed Hash Table*) (43) o Secure-i3 (44).

VALORACIÓN DEL PROYECTO

Este proyecto ha constado de cuatro tareas principales que describiremos brevemente a continuación. Se incluye una estimación temporal de cada una de ellas en días y horas, calculando dos horas de trabajo efectivo por día:

1. **Documentación:** en esta fase se realizó el estudio de la documentación disponible sobre los distintos aspectos tratados en este trabajo. Esto incluye, no sólo al protocolo HIP, sino una amplia variedad de otros protocolos y tecnologías relacionadas. La relación de los principales documentos utilizados en esta tarea puede encontrarse en el apartado de bibliografía (100 días/200 horas)
2. **Pruebas fase I:** son las pruebas realizadas sobre máquinas virtuales que supusieron la primera toma de contacto, instalación y configuración de las implementaciones de HIP. Esta tarea se ha subdividido en otras tres tareas consecutivas, que son:
   2.1. **Planificación:** reflexión previa sobre el tipo de pruebas que se pueden realizar, así como sobre las necesidades de infraestructura para su puesta en marcha (5 días/10 horas).
   2.2. **Instalación:** puesta en marcha del escenario de pruebas y comprobación de su correcto funcionamiento (10 días/20 horas).
   2.3. **Ejecución y resolución de problemas:** donde se realizan las pruebas propiamente dichas. Así mismo, esta tarea incluyó una cierta revisión de las tareas anteriores de planificación e instalación para resolver dificultades imprevistas y fallos del software (20 días/40 horas)
3. **Pruebas Fase II:** son las pruebas realizadas utilizando máquinas reales para los nodos principales de cada escenario. En esta fase se repitieron muchas de las pruebas realizadas en la fase I y se incluyeron algunas nuevas relativas a medidas de rendimiento. Al igual que la fase I esta tarea se subdivide en tres tareas consecutivas:
   3.1. **Planificación** (5 días/10 horas).
   3.2. **Instalación** (10 días/20 horas).
   3.3. **Ejecución y resolución de problemas** (20 días/40 horas).
4. **Elaboracion de la memoria:** cuya misión ha sido la producción de la presente memoria del proyecto (125 días/250 horas).

A continuación, podemos ver un gráfico de Gantt que nos muestra el desarrollo temporal de las tareas descritas anteriormente.

Ilustración 44: diagrama de Gantt del proyecto.

Finalmente, en la siguiente tabla se hace una estimación de los costes del proyecto incluyendo hardware, software y horas de trabajo. Los costes unitarios establecidos son aproximados.

Valoración económica				
Concepto	**Cantidad**	**Coste unidad**	**Coste**	**Observaciones**
Horas de trabajo	620	50	31000	
Software	0	0	0	Todo el software utilizado es libre o con licencia gratuita.
Hardware				
PC Intel Xeon dual core/1.60GHz	1	2000	2000	Servidor de máquinas virtuales.
PC Intel Pentium4/1.50GHz	3	600	1200	Nodos MV, CN y HA/RVS.
Router Cisco 7200 VXR	1	6000	6000	Con soporte para VLANs 802.1q
Conmutador Cisco Catalyst 3750	1	2500	2500	Con soporte para VLANs 802.1q
TOTAL			**42700**	**€**

APÉNDICES

MÁQUINA DE ESTADOS DEL INTERCAMBIO BÁSICO HIP

EL protocolo HIP, en sí mismo, tiene poco estado. En el intercambio básico HIP hay un iniciador y un contestador, y una vez establecida la asociación de seguridad, esta distinción desaparece. La máquina de estados se describe vista desde un único sistema, el cual representa tanto al iniciador como al contestador. Esta máquina de estados se centra en los paquetes I1, R1, I2 y R2 del protocolo HIP; para los mecanismos de movilidad y *multihoming* puede ser necesario introducir nuevos estados.

El siguiente gráfico es una versión simplificada de la máquina de estados, donde sólo se muestran las transiciones más representativas.

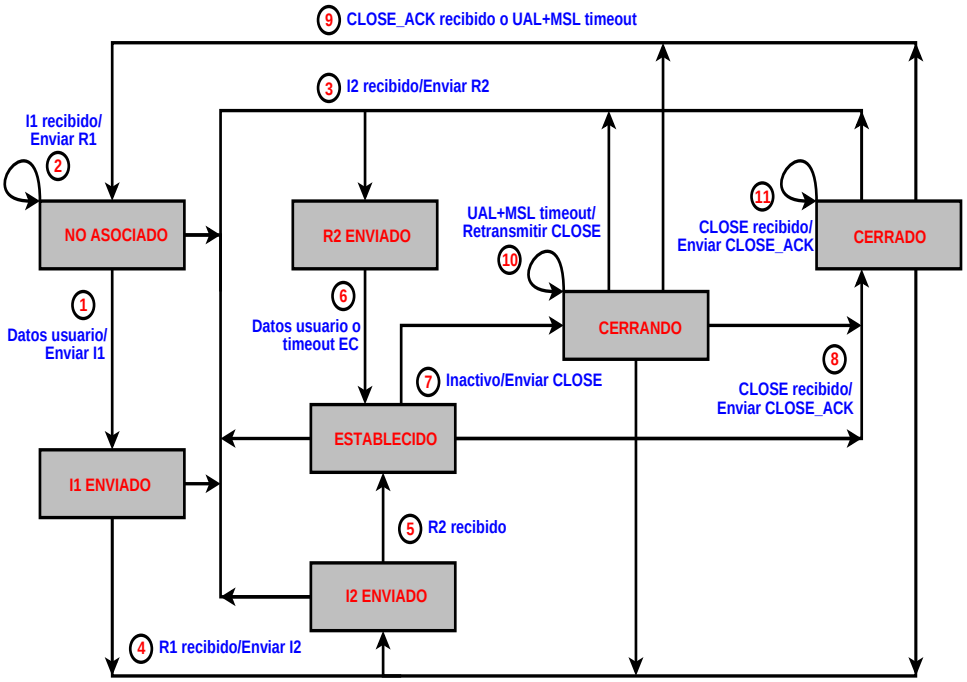

Ilustración 45: máquina de estados del Intercambio Básico HIP.

Los estados tienen la siguiente semántica:

Estado.	Descripción.
NO ASOCIADO	Inicio de la máquina de estados
I1 ENVIADO	Iniciando el intercambio básico HIP.
I2 ENVIADO	Esperando, como iniciador, para completar el intercambio básico HIP.
R2 ENVIADO	Esperando, como contestador, para completar el intercambio básico HIP.
ESTABLECIDO	Asociación HIP completada.
CERRANDO	Cerrando la asociación HIP. No se permite el envío de nuevos datos de usuario.
CERRADO	Asociación HIP cerrada. No se permite el envío de nuevos datos de usuario.
INTERCAMBIO FALLIDO	El intercambio básico HIP ha fallado. La única misión de este estado es esperar un tiempo determinado antes de volver al estado NO ASOCIADO.

A continuación, se hace una descripción más detallada de la máquina de estados por medio de tablas. Los números entre parentesis relacionan algunas de las transiciones con el gráfico anterior.

Tabla de transiciones del estado **NO ASOCIADO**:

Condición.	Transición.
El envío de datos de usuario requiere una nueva asociación HIP.	**Enviar R1** y pasar al estado **I1_ENVIADO** (1).
I1 recibido.	**Enviar R1** permaneciendo en el estado NO_ASOCIADO (2).
I2 recibido y procesado	Si es correcto **enviar R2** y pasar al estado **R2_ENVIADO** (3).
	En caso contrario permanecer en el estado NO_ASOCIADO.
Datos de usuario recibidos de una asociación HIP desconocida.	Permanecer en el estado NO_ASOCIADO. De manera opcional se puede enviar un mensaje ICMP indicativo.
CLOSE recibido.	Permanecer en el estado NO_ASOCIADO. De manera opcional se puede enviar un **mensaje ICMP** indicativo.
Cualquier otro paquete recibido.	**Descartar** y permanecer en el estado NO_ASOCIADO.

Tabla de transiciones del estado **I1 ENVIADO**:

Condición.	Transición.
I1 recibido.	Si el HIT local es más pequeño que el HIT del corresponsal, **descartar** y permanecer en el estado I1_ENVIADO.
	En caso contrario, **enviar R1** y permanecer en el estado **I1_ENVIADO**.
I2 recibido y procesado.	Si es correcto, **enviar R2** y pasar al estado **R2_ENVIADO** (3).
	En caso contrario, permanecer en el estado I1_ENVIADO.
R1 recibido y procesado.	Si es correcto, **enviar I2** y pasar al estado **I2_ENVIADO** (4).
	En caso contrario, permanecer en el estado I1_ENVIADO.
Cualquier otro paquete recibido.	**Descartar** y permanecer en el estado I1_ENVIADO.
Timeout, se incrementa el contador.	Si el contador es menor que el máximo de reintentos (I1_RETRIES_MAX), **Enviar I1** y permanecer en I1_ENVIADO.
	En caso contrario pasar al estado **INTERCAMBIO_FALLIDO**.

Tabla de transiciones del estado **I2 ENVIADO**:

Condición.	Transición.
I1 recibido.	**Enviar R1** y permanecer en el estado I2_ENVIADO.
R1 recibido y procesado.	Si es correcto, **enviar I2** y permanecer en el estado I2_ENVIADO.
	En caso contrario, permanecer en el estado I2_ENVIADO.
I2 recibido y procesado.	Si es correcto, y si el HIT local es más pequeño que el HIT del corresponsal, **descartar** y permanecer en el estado I2_ENVIADO.
	Si es correcto, y si el HIT local es más grande que el HIT del corresponsal, **enviar R2** y pasar al estado **R2_ENVIADO** (3).
	En caso contrario, permanecer en el estado I2_ENVIADO.
R2 recibido y procesado.	Si es correcto, pasar al estado **ESTABLECIDO** (5).
	En caso contrario, permanecer en el estado I2_ENVIADO.
Cualquier otro paquete recibido.	**Descartar** y permanecer en el estado I2_ENVIADO.
Timeout, se incrementa el contador.	Si el contador es menor que el máximo de reintentos (I2_RETRIES_MAX), **enviar I2** y permanecer en I2_ENVIADO.
	En caso contrario pasar al estado **INTERCAMBIO_FALLIDO**.

Tabla de transiciones del estado **R2 ENVIADO**:

Condición.	Transición.
I1 recibido.	**Enviar R1** y permanecer en R2_ENVIADO.
I2 recibido y procesado.	Si es correcto, **enviar R2** y permanecer en el estado R2_ENVIADO.
	En caso contrario, permanecer en el estado R2_ENVIADO.
R1 recibido.	**Descartar** y permanecer en el estado R2_ENVIADO.
R2 recibido.	**Descartar** y permanecer en el estado R2_ENVIADO.
Datos o ACTUALIZACION recibida.	Pasar al estado **ESTABLECIDO** (6).
Timeout de Intercambio Completado.	Pasar al estado **ESTABLECIDO** (6).

Tabla de transiciones del estado **ESTABLECIDO**:

Condición.	Transición.
I1 Recibido.	**Enviar R1** y permanecer en el estado ESTABLECIDO.
I2 recibido, puzle procesado y verificación (posiblemente opaca) de los datos.	Si es correcto, **enviar R2**, descartar asociaciones anteriores y crear una nueva asociación pasando al estado **R2_ENVIADO** (3).
	En caso contrario, permanecer en el estado ESTABLECIDO.
R1 recibido.	**Descartar** y permanecer en el estado ESTABLECIDO.
R2 recibido.	**Descartar** y permanecer en el estado ESTABLECIDO.
Datos recibidos para una asociación HIP.	**Procesar** y permanecer en el estado ESTABLECIDO.
Ningún paquete enviado/recibido durante UAL minutos.	**Enviar CLOSE** y pasar al estado **CERRANDO** (7).
CLOSE recibido y procesado.	Si es correcto, **enviar CLOSE_ACK** y pasar al estado **CERRADO** (8).
	En caso contrario, permanecer en el estado ESTABLECIDO.

Tabla de transiciones del estado **CERRANDO**:

Condición.	Transición.
El envío de datos de usuario requiere una nueva instancia de la asociación HIP.	**Enviar I1** y permanecer en el estado CERRANDO.
I1 recibido.	**Enviar R1** y permanecer en el estado CERRANDO.
I2 recibido y procesado.	Si es correcto, **enviar R2** y pasar al estado **R2_ENVIADO** (3).
	En caso contrario, permanecer en el estado CERRANDO.
R1 recibido y procesado.	Si es correcto, **enviar I2** y pasar al estado **I2_ENVIADO** (4).
	En caso contrario, permanecer en el estado CERRANDO.
CLOSE recibido y procesado.	Si es correcto, **enviar CLOSE_ACK** y pasar al estado **CERRADO** (8).
	En caso contrario, permanecer en el estado CERRANDO.
CLOSE_ACK recibido y procesado.	Si es correcto, **eliminar la asociación HIP** y pasar al estado **NO_ASOCIADO** (9).
	En caso contrario, permanecer en el estado CERRANDO.
Cualquier otro paquete recibido.	**Descartar** y permanecer en el estado CERRANDO.
Timeout, se incrementa el acumulador y se reinicia el cronometro.	Si el acumulador es inferior a UAL + MSL minutos, **retransmitir CLOSE** y permanecer en el estado CERRANDO (10).
	En caso contrario, pasar al estado **NO_ASOCIADO** (9).

Tabla de transiciones del estado **CERRADO**:

Condición.	Transición.
El envío de datos de usuario requiere una nueva instancia de la asociación HIP.	**Enviar I1** y permanecer en el estado CERRADO.
I1 recibido.	**Enviar R1** y permanecer en el estado CERRADO.
I2 recibido y procesado.	Si es correcto, **enviar R2** y pasar al estado **R2_ENVIADO** (3).
	En caso contrario, permanecer en el estado CERRADO.
R1 recibido y procesado.	Si es correcto, **enviar I2** y pasar al estado **I2_ENVIADO** (4).
	En caso contrario, permanecer en el estado CERRADO.
CLOSE recibido y procesado.	Si es correcto, **enviar CLOSE_ACK** y permanecer en el estado CERRADO (11).
	En caso contrario, permanecer en el estado CERRADO.
CLOSE_ACK recibido y procesado.	Si es correcto, **eliminar la asociación HIP** y pasar al estado **NO_ASOCIADO** (9).
	En caso contrario, permanecer en el estado CERRADO.
Cualquier otro paquete recibido.	**Descartar** y permanecer en el estado CERRADO.
Timeout (UAL + 2MSL)	**Eliminar la asociación HIP** y pasar al estado **NO_ASOCIADO** (9).

INSTALACIÓN Y CONFIGURACIÓN DEL SOFTWARE HIP

HIPL

La instalación se hace sobre una distribución Debian *ETCH* básica. Antes de empezar necesitaremos instalar varios paquetes entre los que podemos incluir: "*pkg-config*", "*libtool*", "*autoconf*", "*automake*", "*g++",* "*openssl*", "*libssl-dev*", "*libtiff-dev*", "*libxml2-dev*", "*libglib2*.0-*dev*", "*libatk1.0-dev*", "*libpango1.0-dev*". Además, para compilar el interfaz grafico de HIPL necesitamos la versión 2.10 de "*libgtk2*", sin embargo, en el momento de hacer este trabajo la última versión precompilada para *Debian* es la 2.8, por lo tanto es necesario instalarlo desde su código fuente:

```
# wget http://ftp.gnome.org/pub/gnome/sources/gtk+/2.10/gtk+-2.10.14.tar.bz2
# tar jxvf gtk+-2.10.14.tar.bz2
# cd gtk+-2.10.14
# ./configure --prefix=/usr
# make
# make install
```

A continuación obtendremos el software de HIPL. Este se pueden instalar fácilmente a partir de paquetes pre-compilados tanto para el *kernel* como para los programas en modo usuario; pero dado que se trata de un código experimental e inestable que se actualiza con mucha frecuencia, lo mejor es disponer de la última versión diaria. Para ello el grupo de desarrollo utiliza la herramienta "*tla*", que podemos instalarnos como paquete pre-compilado. Una vez hecho esto, primero nos registramos y después descargamos las fuentes:

```
# tla my-id "Javier Melero <javier@di.uc3m.es>"
# tla register-archive hipl-dev@freelists.org--hipl http://hipl.hiit.fi/hipl/archive/
# tla my-default-archive hipl-dev@freelists.org--hipl
# tla get hipl--main--2.6 hipl--main--2.6
```

A partir de este momento, cada vez que queramos actualizar el código solo tendremos que hacer lo siguiente:

```
# cd hipl--main--2.6/
# tla replay
* tree is already up to date
```

Una vez en el directorio "hip—main--2.6", y antes de comenzar la compilación, hay ejecutar el script "*autogen.sh*", que debe darnos un resultado correcto para poder continuar. Seguidamente compilamos e instalamos:

```
# ./autogen.sh
Generating configure files... may take a while.
configure.ac:8: installing `./missing'
configure.ac:8: installing `./install-sh'
agent/Makefile.am: installing `./depcomp'
Preparing was successful if there was no error messages above.
Now type:
  ./configure && make
NOTE: The commands above only build the userspace apps.
NOTE: You have to build and install the linux kernel separately.
NOTE: You cannot use HIP without applying the interfamily and beet from the patches directory to your kernel!
NOTE: Some features (e.g. firewall ) require './configure --enable-FEATURE'
NOTE: Run './configure --help' for more information
NOTE: libjip and hipsock need to be compiled separately with make

# ./configure && make
# make install
```

A continuación necesitamos instalar un *kernel* que soporte HIP. A partir de la versión 2.6.27 el modo BEET está incluido por defecto, por lo no es necesario su parcheo. Pero, si se desea utilizar alguna versión anterior buscaremos en el árbol de fuentes el directorio "patches/kernel" donde se encuentran parches disponibles para distintas versiones. Tomamos, por ejemplo, la 2.6.23, bajamos sus fuentes, aplicamos todos los parches disponibles para esa versión, compilamos e instalamos. Previamente podemos necesitar instalar el paquete "*kernel-package*".

```
# wget http://www.eu.kernel.org/pub/linux/kernel/v2.6/linux-2.6.23.tar.bz2
# tar jxf linux-2.6.23.tar.bz2
# cd linux-2.6.23
# patch -p1 < /usr/src/hipl--main--2.6/patches/kernel/2.6.23/beet-2.6.23.patch
# cp /boot/config-2.6.18-4-686 .config
# make oldconfig          →TODO POR DEFECTO
# make-kpkg clean
# make-kpkg --initrd --revision=hipl.1.0 kernel_image
# dpkg -i ../linux-image-2.6.23_hipl.1.0_i386.deb
# reboot
```

Cuando el nuevo *kernel* se haya cargado correctamente estaremos listos para empezar a usar HIPL.

Ejecutando el demonio "hipd" por primera vez se generan los HI locales y se almacenan en el directorio "/etc/hip". A continuación, podemos obtener el HIT por defecto mediante el comando:

```
#hipconf get hi default
```

Para añadir la información de la IPv6 y el HIT de coda nodo en su corresponsal se utilizan los ficheros "/etc/hosts" (asociación nombre→IPv6) y "/etc/hip/hosts" (asociación nombre→HIT). El nombre sirve como índice correlativo entre ficheros por lo que debe ser idéntico en ambos. A diferencia de openHIP el demonio de HIPL no intercepta las llamadas a la librería de resolución estándar, por lo que el uso de nombres locales a nivel de aplicación no funciona.

OPENHIP

Los desarrolladores de OpenHIP siguen una política de versiones cerradas con un periodo de duración de varios meses. Por este motivo resulta apropiada la instalación de paquetes pre-compilados en lugar de usar el código fuente. En el momento de hacer este trabajo la versión más reciente es la 0.5.1.

Partiendo de una distribución *Debian ETCH* básica, lo primero que necesitamos hacer es actualizar la versión de "libc6" (la librería c estándar de GNU). El paquete pre-compilado de OpenHIP exige una versión 2.4 o superior de "libc6", mientras que la distribución *ETCH* por defecto instala la 2.3. Para ello editamos el fichero /etc/apt/sources.list y añadimos la línea:

```
deb http://ftp.gul.uc3m.es/debian/ testing main
```

Y a continuación hacemos la actualización:

```
# apt-get update
# apt-get install libc6
```

Ahora ya podemos descargar e instalar el software de OpenHIP. Se ha probado a instalar HIPL y OpenHIP sobre la misma máquina y aparentemente ambas instalaciones no colisionan.

```
# wget http://downloads.sourceforge.net/openhip/openhip_0.5_i386.deb
# dpkg -i openhip_0.5_i386.deb
```

Durante la instalación se genera una HI RSA, así como los ficheros de configuración por defecto. También se pueden generar posteriormente mediante le comando “hitgen”. Una fuente de confusión es el hecho de que estos ficheros se almacenan en el directorio “/usr/local/etc/hip”, sin embargo, el demonio “hip” lee estos ficheros del directorio “/etc/hip”, por lo que es preciso copiarlos antes de su arranque.

```
#cd /usr/local/sbin
#./hitgen -conf
#./hitgen –type RSA –name hip01 –bits 1024
#./hitgen -publish
#cp ../etc/hip /etc/hip
```

A continuación, debemos añadir en el fichero “known_host_identities.xml” de cada nodo los datos de la IPv6 y el HIT de su corresponsal. Como ejemplo el fichero para el nodo hip03 será:

```
<?xml version="1.0" encoding="UTF-8"?>
<known_host_identities>
 <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
  <name>hipserver.mct.phantomworks.org-1024</name>
  <addr>130.76.43.74</addr>
  <HIT>2001:14:4dcd:2a09:74a:caee:2a0:ec4a</HIT>
  <LSI>1.230.120.200</LSI>
 </host_identity>
 <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
  <name>hip01-1024</name>
  <addr>2001:720:410:b131:204:75ff:fefe:2977</addr>
  <HIT>2001:1e:5fc:f9ae:c925:bd28:9275:689</HIT>
  <LSI>1.117.6.137</LSI>
 </host_identity>
</known_host_identities>
```

La información del HIT la encontraremos en el fichero "hip01_host_identities.pub.xml" en el nodo hip01.

HIP4INTER

Es necesario tener instalada la versión exacta de *FreeBSD* que indican los implementadores, aunque ésta sea un poco anticuada. En el momento de hacer este trabajo la versión utilizada es la 6.3, aunque la última versión estable distribuida por *FreeBSD* sea ya la 7.0.

La instalación se hace mediante un paquete que incluye tanto el *kernel* modificado como las aplicaciones de usuario. Se descarga el paquete de la pagina HIP4INTER, se descomprime, y se ejecuta un *script* del directorio "hip/hip4inter" llamado "install.sh". Dicho *script* compila un nuevo *kernel* además de los programas de usuario, por lo puede tardar un buen rato. Después instala todo en su sitio, de modo que solo hay que rebotar la máquina y ya estará lista para usar HIP4INTER.

```
# wget http://hip4inter.net/download/hip_freebsd_20080817.tar.gz
# tar zxf hip_freebsd_20080817.tar.gz
# cd hip/hip4inter
# ./install.sh
# reboot
```

El mismo proceso de instalación realiza la generación del HI local y se almacena en el directorio "/etc/hip", pero pueden generarse otros nuevos con la herramienta "*hip-keygen*". Para la resolución del HIT del contestador en una IP inicial, hay que editar previamente el fichero "/etc/hosts" añadiendo dos instancias para cada nodo, una con su IP y otra con su HIT.

INSTALACIÓN DE ELEMENTOS AUXILIARES

MIPL (*MOBILE IP FOR LINUX*)

La última versión disponible de MIPL solo está disponible para el *kernel Linux* 2.6.16. Una versión algo anticuada, por lo tendremos descargarla, parchear y compilar.

```
# wget http://www.eu.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.tar.bz2
# tar jxf linux-2.6.16.tar.bz2
# wget http://www.mobile-ipv6.org/software/download/mipv6-2.0.2-linux-2.6.16.patch.gz
# gzip -d mipv6-2.0.2-linux-2.6.16.patch.gz
# cd linux-2.6.16
# patch -p1 < ../mipv6-2.0.2-linux-2.6.16.patch
# make menuconfig      → Networking/Networking options:
                              IPv6: advanced router
                              IPv6: policy routing
                              IPv6: source address based routing
                              IPv6: Mobility
                              IPv6: Moblity Debug Message

# make-kpkg clean
# make-kpkg --initrd --revision=mipl.1.0 kernel_image
# dpkg -i ../linux-image-2.6.16_mipl.1.0_i386.deb
# reboot
```

A continuación, tenemos que instalar los programas de espacio usuario. Para ello descargamos las fuentes y compilamos.

```
# wget http://www.mobile-ipv6.org/software/download/mipv6-2.0.2.tar.gz
# tar zxf mipv6-2.0.2.tar.gz
# cd mipv6-2.0.2
# CPPFLAGS='-isystem /usr/src/linux/include' ./configure
# make install
```

Para configurar MIPL se utiliza únicamente el fichero "/usr/local/etc/mip6d.conf". El subdirectorio "extras" incluye ejemplos de configuración para los tres posibles casos de uso: CN, MN y HA.

CONFIGURACIÓN DE *ROUTERS*

Si se desea utilizar un PC con SO Linux como *router* de acceso, es necesario instalar algún software adicional. Lo mismo sucede con el PC que realice el trabajo de *Home Agent* para MIPL, a diferencia del servidor RVS para HIPL que no precisa instalar ningún otro software adicional.

Como software de *router* para Linux se utilizó el paquete *Quagga* que proporciona un interfaz sencillo para configurar las funciones de reenvío de paquetes. Este, además, incluye el protocolo RIPNG (protocolo de *routing* RIP para IPv6) que permite el intercambio de rutas con otros *routers*.

Por otro lado, para la función de envío de los anuncios RA se instaló el paquete "radvd". Aunque estos anuncios también podrían haberse realizado con "quagga" había dos motivos para no hacerlo. Primero, la parametrización de los anuncios en "radvd" es mucho más fiel a la RFC 4861 (41) que "quagga", cuyos comandos de configuración siguen la sintaxis de *Cisco IOS*. Y segundo, MIPL exige tener "radvd" instalado para poder funcionar como *Home Agent*. A continuación, vemos un ejemplo del fichero de configuración "/etc/radvd.conf", indicando los parámetros examinados en las pruebas de movilidad.

```
interface eth1
{
        AdvSendAdvert on;
        AdvIntervalOpt on;              → Permite usar intervalos más bajos
        MaxRtrAdvInterval 0.7;          → Tiempo máximo entre anuncios
        MinRtrAdvInterval 0.3;          → Tiempo mínimo entre anuncios
        AdvDefaultLifetime 1;           → Validez del anuncio (ruta por defecto)
        AdvHomeAgentFlag on;            → Necesario para MIP
        HomeAgentLifetime 10000;        → Idem
        HomeAgentPreference 20;         → Idem
        AdvHomeAgentInfo on;            → Idem
        prefix 2001:720:410:B500::2/64
        {
                AdvRouterAddr on;
                AdvOnLink on;
                AdvAutonomous on;
                AdvValidLifetime 1;     → Validez del prefijo (dirección IP y ruta
local)
                AdvPreferredLifetime 1;
        };
}
```

En un *router Cisco* el intervalo entre anuncios se establece de forma directa, a pesar de lo que prescribe la RFC 4861 (41). También se podría usar un *router Cisco* como HA, pero es necesaria una versión de *IOS* 12.3(14)T ó 12.4. El *router* disponible para estas pruebas tenía una versión de *IOS* anterior y por tanto no disponía de soporte HA para IPv6. A continuación, vemos un ejemplo de configuración.

```
interface GigabitEthernet2/1.2
        description subred B069
        encapsulation dot1Q 11
        ipv6 address 2001:720:410:B069::2/64
        ipv6 nd ra-interval msec 500      → Tiempo medio entre anuncios
        ipv6 nd ra-lifetime 1             → Validez del anuncio (ruta por defecto)
        ipv6 nd prefix default 2 2        → Validez del prefijo (dirección IP y ruta local)
```

RESULTADOS DE TRANSFERENCIAS FTP

La siguiente tabla muestra los resultados detallados de las pruebas de velocidad de transferencia con FTP. Se realizaron diez medidas del tiempo de transferencia de un fichero de 500 Mbytes obteniendo el promedio para cada uno de los siguientes escenarios: una conexión IPv6 simple, una conexión sobre un túnel ESP modo transporte y una conexión sobre HIP. La tabla se acompaña de una grafica que representa las series de datos sobre la función de densidad de la distribución normal, obtenida a partir de su media y desviación estándar respectivas.

Velocidad de Transferencia (MBytes/s)			
Muestra	IPv6	ESP	HIP
1	10,47	9,51	8,50
2	10,36	9,25	8,37
3	10,44	9,27	8,59
4	10,18	9,21	8,61
5	10,47	9,19	8,79
6	10,83	9,37	8,39
7	10,45	9,39	8,58
8	10,49	9,13	8,69
9	10,82	9,11	8,73
10	10,42	9,24	8,70
MÍNIMO	10,18	9,11	8,37
MÁXIMO	10,83	9,51	8,79
MEDIA ACOTADA	10,46	9,26	8,61
PROMEDIO	10,49	9,27	8,60
DESVIACIÓN STD.	0,19	0,12	0,14
PERDIDA DE RENDIMINETO (%)	0	11,49	17,64

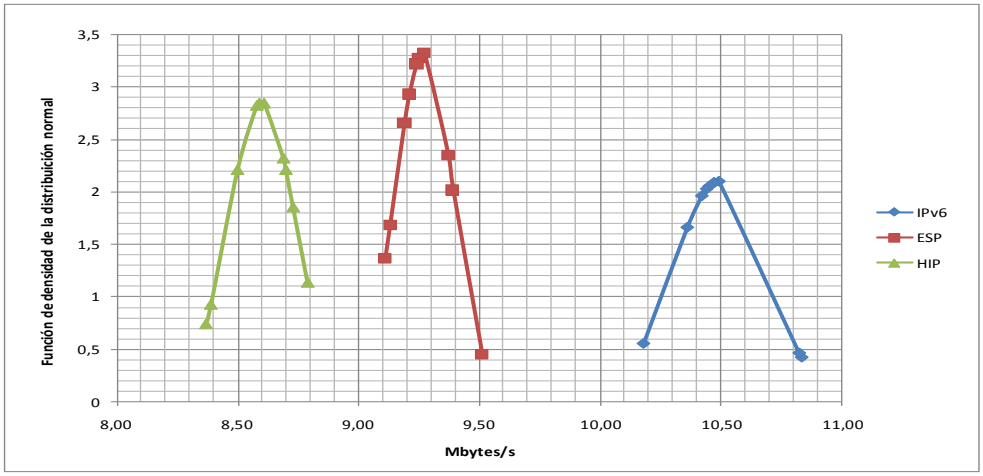

Ilustración 46: muestreo de velocidad de transferencia con FTP.

RESULTADOS DE LA COMPARATIVA MIP/HIP

Cada tabla muestra los resultados obtenidos con diferentes tiempos medios entre anuncios RA (60ms., 1s. y 3s.). Cada prueba consta de veinte movimientos usando HIP o MIP como soporte para la movilidad. MIP puede usarse con cuatro posibles configuraciones, según se utilicé *eager switching* (ES) o *lazy switching* (LS) combinado con la opción de *route optimization* activada o no (+/-RO). HIP sólo se prueba con *eager switching*.

Cada tabla se acompaña con una grafica que representa las series de datos sobre la función de densidad de la distribución normal, obtenida a partir de su media y desviación estándar respectivas.

TIEMPO DE TRASPASO con un RA cada 60ms.										
Muestra	MIP (LS, -RO)		MIP (LS, +RO)		MIP (ES, -RO)		MIP (ES, +RO)		HIP (ES)	
	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibodos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimados (s)
1	149	2,55	147	2,65	170	1,50	164	1,80	169	1,55
2	155	2,25	146	2,70	168	1,60	175	1,25	171	1,45
3	151	2,45	146	2,70	173	1,35	178	1,10	175	1,25
4	146	2,70	141	2,95	169	1,55	171	1,45	165	1,75
5	157	2,15	145	2,75	159	2,05	164	1,80	162	1,90
6	151	2,45	151	2,45	173	1,35	174	1,30	173	1,35
7	159	2,05	148	2,60	174	1,30	170	1,50	173	1,35
8	143	2,85	145	2,75	172	1,40	167	1,65	166	1,70
9	153	2,35	150	2,50	166	1,70	173	1,35	168	1,60
10	141	2,95	148	2,60	175	1,25	170	1,50	174	1,30
11	143	2,85	144	2,80	165	1,75	166	1,70	177	1,15
12	143	2,85	147	2,65	169	1,55	177	1,15	170	1,50
13	152	2,40	148	2,60	166	1,70	163	1,85	170	1,50
14	142	2,90	143	2,85	158	2,10	166	1,70	166	1,70
15	153	2,35	158	2,10	171	1,45	163	1,85	164	1,80
16	151	2,45	150	2,50	173	1,35	168	1,60	174	1,30
17	149	2,55	138	3,10	165	1,75	169	1,55	169	1,55
18	154	2,30	151	2,45	162	1,90	161	1,95	173	1,35
19	157	2,15	153	2,35	172	1,40	161	1,95	164	1,80
20	152	2,40	155	2,25	166	1,70	177	1,15	175	1,25
Mínimo		2,05		2,10		1,25		1,10		1,15
Máximo		2,95		3,10		2,10		1,95		1,90
Media		2,50		2,62		1,59		1,56		1,51
Media Acotada		2,47		2,63		1,56		1,58		1,49
Desviación Estándar		0,27		0,23		0,24		0,27		0,22

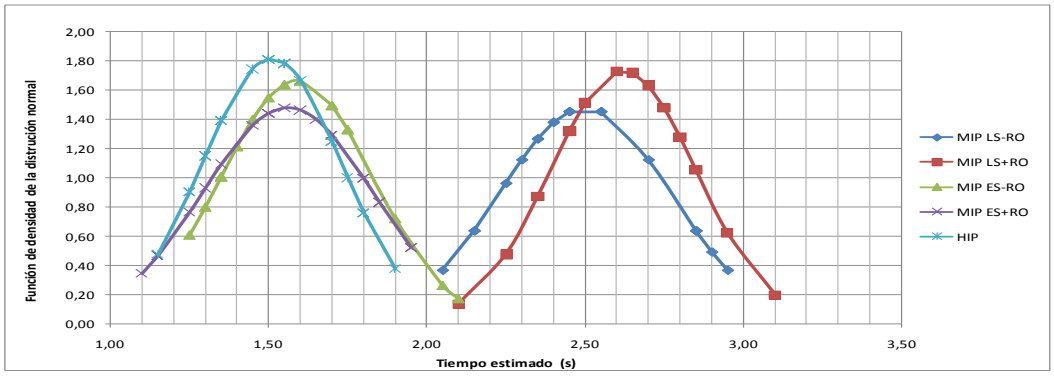

Ilustración 47: muestreo de tiempo de traspaso con un RA cada 60ms.

TIEMPO DE TRASPASO con un RA cada 1s.										
Muestra	MIP (LS, -RO)		MIP (LS, +RO)		MIP (ES, -RO)		MIP (ES, +RO)		HIP (ES)	
	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibodos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimados (s)
1	143	2,85	118	4,10	127	3,65	144	2,80	158	2,10
2	138	3,10	137	3,15	162	1,90	157	2,15	163	1,85
3	143	2,85	107	4,65	153	2,35	164	1,80	157	2,15
4	141	2,95	130	3,50	142	2,90	154	2,30	138	3,10
5	119	4,05	138	3,10	158	2,10	165	1,75	157	2,15
6	143	2,85	143	2,85	162	1,90	167	1,65	164	1,80
7	132	3,40	135	3,25	147	2,65	158	2,10	153	2,35
8	138	3,10	137	3,15	155	2,25	167	1,65	131	3,45
9	106	4,70	149	2,55	160	2,00	142	2,90	145	2,75
10	126	3,70	125	3,75	125	3,75	133	3,35	152	2,40
11	128	3,60	128	3,60	156	2,20	126	3,70	163	1,85
12	135	3,25	127	3,65	163	1,85	140	3,00	171	1,45
13	149	2,55	123	3,85	150	2,50	159	2,05	137	3,15
14	144	2,80	142	2,90	120	4,00	141	2,95	156	2,20
15	136	3,20	143	2,85	156	2,20	119	4,05	131	3,45
16	134	3,30	138	3,10	166	1,70	136	3,20	169	1,55
17	137	3,15	134	3,30	138	3,10	130	3,50	130	3,50
18	136	3,20	105	4,75	163	1,85	160	2,00	128	3,60
19	126	3,70	146	2,70	155	2,25	169	1,55	134	3,30
20	140	3,00	117	4,15	109	4,55	163	1,85	143	2,85
Mínimo		2,55		2,55		1,70		1,55		1,45
Máximo		4,70		4,75		4,55		4,05		3,60
Media		3,27		3,45		2,58		2,52		2,55
Media Acotada		3,17		3,36		2,34		2,41		2,52
Desviación Estándar		0,50		0,62		0,82		0,77		0,70

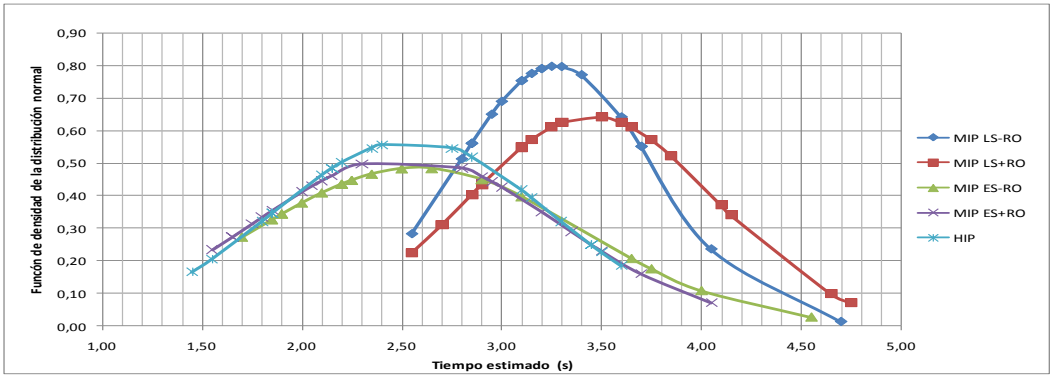

Ilustración 48: muestreo de tiempo de traspaso con un RA cada 1s.

TIEMPO DE TRASPASO con un RA cada 3s.										
Muestra	MIP (LS, -RO)		MIP (LS, +RO)		MIP (ES, -RO)		MIP (ES, +RO)		HIP (ES)	
	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibodos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimado (s)	Paquetes Recibidos	Tiempo Estimados (s)
1	43	7,85	111	4,45	104	4,80	109	4,55	101	4,95
2	123	3,85	116	4,20	102	4,90	135	3,25	104	4,80
3	130	3,50	117	4,15	109	4,55	111	4,45	106	4,70
4	55	7,25	137	3,15	124	3,80	95	5,25	148	2,60
5	126	3,70	114	4,30	94	5,30	90	5,50	161	1,95
6	81	5,95	109	4,55	125	3,75	97	5,15	104	4,80
7	63	6,85	139	3,05	155	2,25	159	2,05	131	3,45
8	129	3,55	109	4,55	28	8,60	105	4,75	124	3,80
9	118	4,10	30	8,50	157	2,15	129	3,55	79	6,05
10	101	4,95	107	4,65	109	4,55	107	4,65	112	4,40
11	138	3,10	96	5,20	143	2,85	137	3,15	126	3,70
12	113	4,35	16	9,20	117	4,15	79	6,05	95	5,25
13	83	5,85	75	6,25	153	2,35	98	5,10	106	4,70
14	52	7,40	96	5,20	87	5,65	102	4,90	91	5,45
15	95	5,25	71	6,45	117	4,15	100	5,00	128	3,60
16	70	6,50	140	3,00	109	4,55	117	4,15	90	5,50
17	98	5,10	72	6,40	94	5,30	141	2,95	148	2,60
18	50	7,50	112	4,40	119	4,05	91	5,45	86	5,70
19	123	3,85	93	5,35	119	4,05	147	2,65	114	4,30
20	64	6,80	65	6,75	109	4,55	93	5,35	176	1,20
Mínimo		3,10		3,00		2,15		2,05		1,95
Máximo		7,85		9,20		8,60		6,05		6,05
Media		5,36		5,19		4,32		4,40		4,18
Media Acotada		5,27		4,89		4,32		4,63		4,38
Desviación Estándar		1,57		1,66		1,43		1,10		1,31

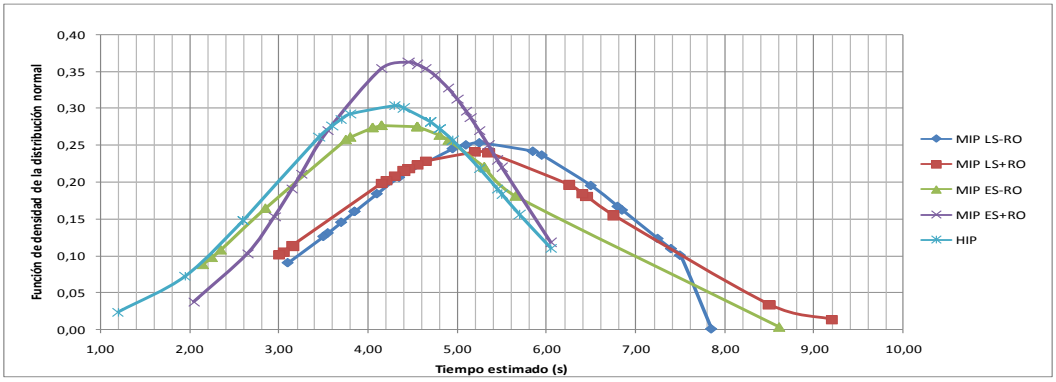

Ilustración 49: muestreo de tiempo de traspaso con un RA cada 3s.

PARCHE PARA LA MODIFICACIÓN DE RADVDUMP

El fichero del parche se encuentra el directorio “patches/misc” del árbol de fuentes de HIPL. Para modificar la herramienta *radvdump* se deben obtener las fuentes del paquete *radvd-1.1* y, desde el directorio superior, ejecutar el comando:

```
hip01:/usr/src# patch -p0 < radvd-1.1_radvdump_eager_switching_melero.patch
(Stripping trailing CRs from patch.)
patching file radvd-1.1/radvdump.c
```

Seguidamente, se compila el paquete de la forma habitual. Es necesario, además, tener instalado el paquete *iproute* (*IPROUTE2 utility suite*) que contiene los comandos para el manejo de las tablas de direcciones y rutas locales.

TABLA DE ILUSTRACIONES

BIBLIOGRAFÍA

1. **Saltzer, J.** *On the Naming and Binding of Network Destinations.* RFC 1498, 1993.

2. *Endpoints and Endpoints Names: A Proposed Enhancement to the Internet Architecture.* **Chiappa, N.** draft, 1999.

3. **Lear, E. and Droms, R.** *What's In A Name: Thoughts from the NSRG.* draft-irtf-nsrg-report-10, 2003.

4. **Moskowitz, R. and Nikander, P.** *Host Identity Protocol (HIP) Architecture.* RFC-4423, 2006.

5. **Johnson, D., Perkins, C. and Arkko, J.** *Mobility Support in IPv6.* RFC-3775, 2004.

6. **Narayanan, S.** *Detecting Network Attachment in IPv6 Networks (DNAv6).* draft-ietf-dna-protocol-08, Julio 2008.

7. **Kent, S. and Seo, K.** *Security Architecture for the Internet Protocol.* RFC-4301, 2005.

8. **Perkins, C.** *IP Mobility Support for IPv4.* RFC-3344, 2002.

9. **Stewart, R.** *Stream Control Transmission Protocol.* RFC-4960, 2007.

10. **Kaufman, C.** *Internet Key Exchange (IKEv2) Protocol.* RFC-4306, 2005.

11. **Eronen, P.** *IKEv2 Mobility and Multihoming Protocol (MOBIKE).* RFC-4555, 2006.

12. **Borella, M., et al.** *Realm Specific IP: Framework.* RFC-3102, 2001.

13. **Srisuresh, P., et al.** *Middlebox Communication Architecture and Framework.* RFC-3303, 2002.

14. **Cheriton, D. and Gritter, M.** *TRIAD: A Scalable Deployable NAT-based Internet Architecture.* 2000.

15. **Francis, P. and Gummadi, R.** *IPNL: A NAT Extended Internet Architecture.* 2001.

16. **Brander, S., Markin, A. and Schiller, J.** *A Framework for Purpose Built Keys.* draft-bradner-pbk-frame-06, 2003. draft-bradner-pbk-frame-06.

17. **Moskowitz, R., et al.** *Host Identity Protocol.* RFC-5201, Abril 2008.

18. **Eastlake, D.** *RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS).* RFC-3110, Mayo 2001.

19. **EastLake, D.** *DSA KEYs and SIGs in the Domain Name System (DNS).* RFC-2536, 1999.

20. **Nikander, P., Laganier, J. and Dupont, F.** *An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID).* RFC-4843, 2007.

21. **Aura, T.** *Cryptographically Generated Addresses (CGA).* RFC-3972, 2005.

22. **Eastlake, D. and Hansen, T.** *US Secure Hash Algorithms (SHA and HMAC-SHA).* RFC-4634, 2006.

23. **Hinden, R. and Deering, S.** *IP Version 6 Addressing Architecture.* RFC-4291, 2006.

24. **Nikander, P. and Laganier, J.** *Host Identity Protocol (HIP) Domain Name System (DNS) Extension.* RFC-5205, Abril 2008.

25. **Komu, M. and Henderson, T.** *Basic Socket Interface Extensions for Host Identity Protocol (HIP).* draft-ietf-hip-native-api-05, Julio 2008.

26. **Jokela, P., Moskowitz, R. and Nikander, P.** *Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP).* RFC-5202, Abril 2008.

27. **Kent, S.** *IP Encapsulating Security Payload (ESP).* RFC-4303, 2005.

28. **Nikander, P. and Melen, J.** *A Bound End-to-End Tunnel (BEET) mode for ESP.* draft-nikander-esp-beet-mode-09, Agosto 2008.

29. **Nikander, P., et al.** *End-Host Mobility and Multihoming with the Host Identity Protocol.* RFC-5206, Abril 2008.

30. **Laganier, J. and Eggert, L.** *Host Identity Protocol (HIP) Rendezvous Extension.* RFC-5204, Abril 2008.

31. **Laganier, J., Koponen, T. and Eggert, L.** *Host Identity Protocol (HIP) Registration Extension.* RFC-5203, Abril 2008.

32. **Tschofenig, H., Shanmugam, M. and Stiemerling, M.** *Traversing HIP-aware NATs and Firewalls: Problem Statement and Requirements.* draft-tschofenig-hiprg-hip-natfw-traversal-06, 2007.

33. **Carpenter, B. and Brim, S.** *Middleboxes: Taxonomy and Issues.* RFC-3234, 2002.

34. **Melen, J., Ylitalo, J. and Salmela, P.** *Security Parameter Index multiplexed Network Address Translation (SPINAT).* draft-melen-spinat-01, Julio 2008.

35. **Huttunen, A., et al.** *UDP Encapsulation of IPsec ESP Packets.* RFC-3948, 2005.

36. **Kivinen, T., et al.** *Negotiation of NAT-Traversal in the IKE.* RFC-3947, 2005.

37. **Stiemerling, M., Quittek, J. and Eggert, L.** *NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication.* RFC-5207, Abril 2008.

38. **Komu, M., et al.** *Basic HIP Extensions for Traversal of Network Address Translators.* draft-ietf-hip-nat-traversal-05, Octubre 2008.

39. **Rosenberg, J.** *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols.* draft-ietf-mmusic-ice-19, 2007.

40. **Nordmark, E. and Bagnulo, M.** *Shim6: Level 3 Multihoming Shim Protocol for IPv6.* draft-ietf-shim6-proto-11, Diciembre, 2008.

41. **Narten, T., et al.** *Neighbor Discovery for IP Version 6 (IPv6).* RFC-4861, 2007.

42. **Jiang, Sheng.** *Hierarchical Host Identity Tag Architecture.* draft-jiang-hiprg-hhit-arch-01, Octubre 2008.

43. **Nikander, P., Arkko, J. and Ohlman, B.** *Host Identity Indirection Infrastructure (Hi3).* draft-nikander-hiprg-hi3-00, 2007.

44. **Ahrenholz, J.** *HIP DHT Interface.* draft-ahrenholz-hiprg-dht-03, Octubre 2008.

45. **Arkko, J. and van Beijnum, I.** *Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming.* draft-ietf-shim6-failure-detection-13, Junio 2008.

46. *End-Host Mobility and Multihoming with the Host Identity Protocol.* **RFC-5206.** Abril 2008.

47. *Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP).* **RFC-5202.** Abril 2008.

48. **O'Dell, M.** *GSE, An Alternate Addressing Architecture for IPv6.* draft-ietf-ipngwg-gseaddr-00, 1997.

49. **Farinacci, D., et al.** *Locator/ID Separation Protocol (LISP).* draft-farinacci-lisp-12.txt, Marzo 2009.